Nebulas: Using Distributed Voluntary Resources to Build Clouds

Abhishek Chandra and Jon Weissman
Department of Computer Science and Engineering
University of Minnesota
Minneapolis, MN 55455
{*chandra, jon*}*@cs.umn.edu*

Abstract

Current cloud services are deployed on well-provisioned and centrally controlled infrastructures. However, there are several classes of services for which the current cloud model may not fit well: some do not need strong performance guarantees, the pricing may be too expensive for some, and some may be constrained by the data movement costs to the cloud. To satisfy the requirements of such services, we propose the idea of using distributed voluntary resources—those donated by end-user hosts—to form *nebulas:* more dispersed, less-managed clouds. We first discuss the requirements of cloud services and the challenges in meeting these requirements in such voluntary clouds. We then present some possible solutions to these challenges and also discuss opportunities for further improvements to make nebulas a viable cloud paradigm.

1 Introduction

In the cloud computing domain, a cloud signifies a service provided to a user that hides details of the actual location of the infrastructure resources from the user. Most currently deployed clouds [9, 14, 19, 2] are built on a well-provisioned and well-managed infrastructure, such as a data center, that provides resources and services to users. The underlying infrastructure is typically owned and managed by the cloud provider (e.g., Amazon, IBM, Google, Microsoft, etc.), while the user pays a certain price for their usage of the resources. There is also the notion of strong resource and/or performance guarantees between the cloud provider and the user, that ensures that the user sees the performance they expect to see. Cloud services tend to fall into several categories: long-term state and data storage [23], "one-shot" burst of computation [16], and interactive end user-oriented services [15].

While the current cloud infrastructures are important for the ease of use and performance they provide to their users, there are several classes of services for which the current cloud model may not fit well. We describe three such classes and give an illustrative example for each:

• *Experimental cloud services:* These are services that may eventually get deployed on a production system (which itself can be a commercial cloud). However, before the actual deployment, the service developers may want to "test drive" the service to make it production-ready (e.g., remove bugs), or to test it for its viability as a cloud service (e.g., gauge user demand/popularity). To carry out such an experimental deployment, they may need sustained access to a large-scale "test cloud" which can provide a realistic deployment environment without the costs and robustness of a production setting.

Example: A group of entrepreneurial Computer Scientists, concerned about the state of Computer Science research, decide to develop a research quality assurance service that could be employed by conferences and journals to ensure the novelty and originality of publications. As part of this service, submitted papers can be compared against an archive of already published papers to check for several problems ranging from similar results/text appearing in multiple papers to full-scale plagiarism. However, they may wish to understand the potential popularity and accuracy as well as the resource demands of such a service before partnering with organizations like ACM and IEEE to host it on a commercial cloud.

• *Dispersed-Data-intensive services:* These are services which rely on large amounts of dispersed data, and where moving data to a centralized cloud can be prohibitively expensive and inefficient. In this case, it would be preferable to move the computational resources closer to the data while providing sufficient computational capability.

Example: A group of social scientists wish to provide a service that would analyze a large number of geographically distributed user blogs containing text, audio, and video content to find interesting social trends. Given the large number and size as well as the distributed nature of such blogs, it would be important to move the analysis service closer to where the blogs are actually stored.

• *Shared services:* These services would be provided by users or organizations that wish to freely share their own private applications with others as a "public service", but require deployment resources (e.g., computational resources or network bandwidth). However, since these services may not be commercial, the service deployers may not want to pay the cost for running the services. At the same time, these services may need arbitrary scale-up/scale-down based on user demand.

Example: A user has built a custom "tour" of his recent trip to Paris including maps, images, video, commentary, etc. This tour is also context-aware: given a user's GPS coordinates, locations and information pop up. Since this service with its data is large in size, it would need a large amount of network bandwidth to serve the content to interested users. Moreover, it may need context-aware processing or data fetching for better user experience, which would be hard to support on the user-end if the user is using a thin client such as a PDA or mobile phone.

Many of the above services may have weak performance and robustness requirements, so that paying for stringent requirements of high availability, accuracy, and performance, as provided by many current cloud providers may be unnecessary and undesirable.

To host such services, we propose the notion of *nebulas:* more dispersed, less managed clouds, constructed using voluntary resources—those donated by end-user hosts—such as those used in @home systems [1] and P2P systems [24, 26, 13]. Nebulas draw on many of the ideas advanced in P2P systems, Grids [11, 1], and distributed data centers [8]. We believe nebulas are fully complementary to commercial clouds and in some cases may represent a transition pathway. Volunteer resources are attractive for several reasons:

• *Scalability:* Many existing volunteer platforms consist of millions of hosts and users, providing a large amount of resource capacity and scalability. E.g.: Folding@home [10] has approximately 250K hosts providing over 1 Petaflop, which is comparable to some of the fastest supercomputers in the world today. Kazaa [24] has an average of 3.5M users representing several Terabits/sec of aggregate bandwidth.

• *Dispersion:* Volunteer nodes are likely to be geographically distributed. This can enable better mapping of services to resources, in order to reduce data movement costs, as well as to provide end-user-specific, context-aware service deployment.

• *Low cost of deployment:* Volunteer resources are basically available for free or at very low cost. These are simply idle resources already available in the system. They do not impose any additional hardware, maintenance, or energy costs, beyond what they are already using.

Despite these attractive properties, most existing applications deployed on volunteer systems are largely best-effort, and these systems would not meet many requirements that define clouds today. In this paper, we outline the opportunities and challenges of using such voluntary resources to build nebulas, and present the key issues that would need to be solved to make them suitable for use for the above service classes. We also detail how existing point solutions could be synthesized to realize nebulas. In this paper, we mainly focus on performance and reliability issues for nebulas, and omit discussion on some other issues due to space limitations. For instance, we do not address the problem of incentivizing donation, for which techniques such as credit systems and market economy models have been proposed in the volunteer computing space [4, 28]. Similarly, we do not discuss deployment issues such as interfaces, APIs, and sandboxing/isolation mechanisms such as virtualization, which will depend on the specific nature of emerging services and resource owners. Finally, we omit security as a specific requirement for nebulas, because we believe highly secured cloud applications are unlikely to use the nebula approach (other than for non-secured testing).

Differences from existing volunteer platforms: Before discussing the opportunities and challenges involved in building nebulas, we first outline the major differences we see between nebulas and existing volunteer platforms:

• *Cross-component interactions:* Most existing @home systems are designed for embarrassingly parallel computations where there is no interaction between the different compute tasks. However, for services running on a nebula, tasks belonging to a user request would have much tighter coupling and interactions as well as collective performance goals, requiring careful allocation of localized nodes for their execution.

• *Locality- and context-awareness:* While existing @home and P2P systems do not distinguish between different nodes in the systems for allocating compute tasks and data files, in a nebula, particularly for dispersed-data-intensive and shared services, the data-computation locality, as well as the knowledge of the user context (their location, device capacity, etc.) would be critical in making resource allocation decisions.

• *Dynamic state maintenance:* Many cloud services are likely to be stateful, and this state would need to be maintained across failures and churn likely in a volunteer platform. As opposed to @home applications, where computations are largely stateless and can be re-executed easily, nebulas will need to maintain distributed shared state, that can be easily retrieved and used by a service.

2 Nebulas: Requirements and Challenges

In this section, we outline some of the requirements that a voluntary infrastructure must satisfy to be acceptable as a viable cloud platform, and outline the challenges that must be solved to meet these requirements.

Requirement 1: Provide Service-Centric Performance Differentiation

To deploy multiple services on a nebula, and to support multiple user requests for a service, the platform must provide differentiation between these different services/user requests, in terms of their service-specific performance metrics, such as response time, throughput, etc.

Traditional voluntary infrastructures such as BOINC [1] are tuned for never-ending best-effort computations that have no completion boundaries. Thus, different units of a computation simply contribute to incrementally building up partial results, and there is no notion of meeting any application-centric performance metric. For such an application type, one does not have to distinguish between different tasks, so that they can be treated equally and little state needs to be maintained about individual tasks.

While this model is appropriate for one-shot bursty computational applications, a hosted cloud service would be characterized by separate requests being submitted by end-users, and these requests would have to be executed concurrently in the cloud. The cloud would then require mechanisms to differentiate between tasks corresponding to different requests and collate their results separately. For instance, the cloud would have to maintain some state regarding the execution status of each individual request (e.g., its component tasks, completed partial results, volunteers allocated to the request, etc.). At the same time, the interaction between the volunteer nodes would require an awareness of request-oriented nature of the hosted service. For instance, work for new requests should be pushed out as soon as possible to reduce the request execution time, if there is a need to bound the request completion times.

Challenges: Meeting service-centric performance requirements is challenging due to the heterogeneous and time-varying behavior of voluntary nodes. As illustration, we deployed a simple cloud service from the domain of Bioinformatics called BLAST [3] on a small shared cloud using PlanetLab. A high degree of heterogeneity was observed (Figure 1), both in terms of the computational capacity and communication bandwidth of the participating nodes. We can expect a nebula based on volunteer nodes to exhibit an even greater degree of variance than a fixed infrastructure such as PlanetLab.

Requirement 2: Couple Data and Computation

Unlike their highly centralized counterparts, nebulas may contain distributed data and computational resources. For cloud services that need to operate on dispersed data, either computation must be moved close to the data, or the data must be moved close to the computation. This is a critical issue as many emerging services are characterized by their dependence on large quantities of data. Hosting such services on voluntary clouds raises several novel challenges. First, decomposing the work into separate decoupled tasks becomes more difficult due to the inherent data dependency of these services. Second, if the data source has to transfer large amounts of data to the volunteer nodes, network bandwidth becomes an important factor in addition to the node's computational resources. Moreover, to avoid large communication overhead, computation may have to be collocated with the data, which could require careful selection of volunteer nodes in close proximity to the data store.

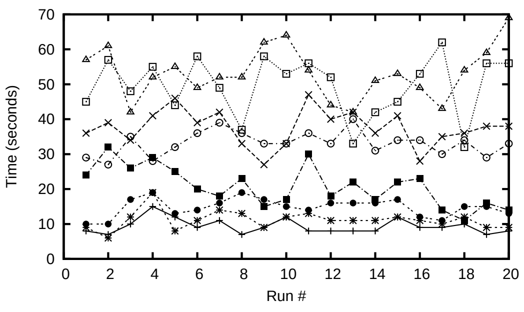
The quality of potential computational resources must be weighed against the cost of data communication to them. Quality can reflect metrics such as performance or reliability/availability. For the case of performance, there may be a tradeoff between the computational capability of a node and its proximity to the data source.

Challenges: The distribution of data and computation pose significant challenges in meeting service goals. In the realm of performance, the heterogeneity and unpredictable nature of network bandwidth makes it difficult to estimate the cost of data communication. Such cost estimation is needed when deciding which compute resources to use. Choosing the compute resources to use from a potentially very large set of volunteers must be done efficiently and a scalable solution is needed.

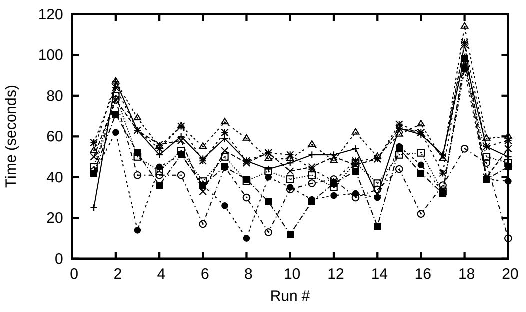
Requirement 3: Provide Robustness to Small-Scale Failures

While a nebula would support fairly weak guarantees on the performance and reliability of a service, it should still provide robustness to small-scale and localized failures. For instance, churn would be high in such systems, and a nebula should ensure that the arrival and departure of individual nodes does not impact the overall functionality of the services deployed on it. In particular, it must be able to preserve service state across failures. Similarly, there may be small-scale "bad" behavior, such as credit-hawking or freeloading, particularly based on the incentive model, and even disruptive or Byzantine behavior on part of some of the participating nodes. However, a nebula should be able to prevent such nodes from subverting or preventing the execution of deployed services. Effectively, a nebula should be able to quarantine such small-scale failures and bad behavior, and be self-organizing to recover from the impact of such nodes on the service performance and reliability.

Challenges: Churn and failures are expected to be the common case in a voluntary platform. Lack of central monitoring and control pose a major challenge in pro-



(a) Computation time



(b) Communication time

Figure 1: The computation and communication time taken by different voluntary nodes over multiple runs.

viding reliability and state-maintenance in the presence of such failures. The reliability of nodes has to be incorporated in service deployment decisions in addition to performance criteria such as computational speed and network bandwidth.

3 Building Nebulas: Possible Solutions

We now outline some of the approaches that can be used to overcome the challenges outlined above. We note that these approaches are only a subset of possible solutions, and many of these challenges are also fertile ground for further research.

Handling Heterogeneity

The diversity of volunteer nodes must be harnessed in a service-specific way. Here, we focus on heterogeneity as it impacts performance. First of all, large-scale resource discovery techniques [18, 22, 5] could be employed to select a suitable set of resources for service deployment. These resources would be selected based on their resource capabilities, and their long-term stability.

Further, service performance metrics would govern how heterogeneity of selected resources is to be handled. We present one such example. Many services attracted to nebulas would be those that require large computations, e.g. large-scale image analysis or scientific computing. Such services are amenable to parallel processing - each service request would be decomposed into separate tasks and run on different volunteers. The response time for each service request can be reduced by smart exploitation of the underlying infrastructure heterogeneity. For instance, since the completion time of the request is dependent on the slowest node, the tasks should be allocated according to individual node capabilities (e.g., by assigning larger tasks to faster nodes with faster communication paths). Thus, tasks may be sized in proportion to the node capacities for load balancing, and may also be decomposed into smaller fixed sizes to account for performance fluctuation [27]. Similar techniques can be used for exploiting heterogeneity in a nebula for providing service-centric performance.

Handling Data-Compute Dependence

Selecting voluntary nodes for task allocation must take into account the location of needed data. The first challenge is to locate data (presuming its location isn't known a-priori). A rich array of techniques exist for finding data by name or metadata in voluntary P2P networks [7, 24, 26]. Once the data locations are known, the network distance from potential volunteer nodes should be considered to meet performance objectives. A wide array of network performance estimation techniques have been proposed [21, 12] that rely on active probes. The downside is that active probes add overhead and consume network resources. An alternative approach is to estimate network performance in a passive manner based on prior data downloads. However, it is unlikely that a specific candidate volunteer (being considered for service deployment) has interacted with a particular data source to have prior measurements. In [20], we have developed a framework called OPEN that can utilize network measurement data from other nodes (obtained via lazy gossip) and adjust those measurements to account for the characteristics of the candidate volunteer. This approach is accurate, low overhead, and leads to good volunteer selections and would be well-suited to the nebula environment.

Handling Failures

Voluntary nodes must be selected not only based on their performance characteristics, but also their reliability, which can be affected by several factors, including: churn caused by node failures as well as revocation of the donated resources, network failures resulting in node disconnections, and misconfigurations or malicious behavior. Replication is a widely-used technique for overcoming failures, used in distributed storage systems [17] and for Byzantine fault tolerance [6]. Since the churn and failure profiles of nodes may vary widely in volunteer platforms, replication techniques will have to incorporate knowledge about individual node reliability values, if it can be determined. For instance, node reliability values can be used to perform dynamic replication [25] in such systems. In this approach, the degree of replication can be varied based on the reliability of participating nodes, in order to achieve service-level reliability metrics without sacrificing performance.

In order to maintain the state of a service across node failures and churn, a nebula may have to support more aggressive checkpointing techniques to avoid losing too much state. In addition, it would need to support a state-maintenance layer for easy storage and retrieval of preserved state by a service. An interesting possibility could be to use a distributed volunteer data storage platform (such as Bittorrent) to maintain and disseminate this state. The tradeoff in the cost of maintaining such state against recreating it (e.g., by recomputation) will depend on the statefulness of the deployed service, as well as the reliability of the volunteer resources.

4 Conclusion

In this paper, we presented the notion of nebulas as an alternative way to construct cloud infrastructures using distributed voluntary resources. These nebulas are geared towards hosting cloud services which may not fit the current well-managed, pay-as-you-go cloud model. We presented the requirements and challenges for hosting such services on volunteer platforms, and then discussed possible solutions to overcome some of these challenges. We believe that nebulas can exist as complementary infrastructures to clouds, and can even serve as a transition pathway for many services that would eventually be hosted on clouds.

References

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

[2] Azure Services Platform. `http://www.microsoft.com/azure/default.mspx`.

[3] The Basic Local Alignment Search Tool (BLAST). `http://www.ncbi.nlm.nih.gov/blast`.

[4] BOINC Stats. `http://boincstats.com`.

[5] M. Cardosa and A. Chandra. Resource Bundles: Using Aggregation for Statistical Wide-Area Resource Discovery and Allocation. In *ICDCS*, June 2008.

[6] M. Castro and B. Liskov. Practical Byzantine Fault Tolerance. In *OSDI*, Feb. 1999.

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

[8] K. Church, A. Greenberg, and J. Hamilton. On Delivering Embarrassingly Distributed Cloud Services. In *HotNets*, 2008.

[9] Amazon Elastic Compute Cloud (Amazon EC2). `http://aws.amazon.com/ec2`.

[10] Folding@home distributing computing project. http://folding.stanford.edu.

[11] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications*, 11(2):115–128, 1997.

[12] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. IDMaps: a global internet host distance estimation service. *ACM Transactions on Networking*, 9(5):525–540, Oct. 2001.

[13] Y. Fu, J. Chase, B. Chun, S. Schwab, and A. Vahdat. SHARP: An Architecture for Secure Resource Peering. In *SOSP*, 2003.

[14] Google App Engine. `http://code.google.com/appengine`.

[15] Google Maps. `http://maps.google.com`.

[16] D. Gottfrid. Self-service, Prorated Super Computing Fun! `http://open.blogs.nytimes.com/2007/11/01/self-service-prorated-super-computing-fun`.

[17] A. Haeberlen, A. Mislove, and P. Druschel. Glacier: Highly Durable, Decentralized Storage Despite Massive Correlated Failures. In *NSDI*, 2005.

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

[19] IBM Cloud Computing. `http://www.ibm.com/ibm/cloud`.

[20] J. Kim, A. Chandra, and J. Weissman. OPEN: Passive Network Performance Estimation for Data-intensive Applications. Technical Report 08-041, Dept. of CSE, Univ. of Minnesota, 2008.

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

[22] D. Oppenheimer, J. Albrecht, D. Patterson, and A. Vahdat. Distributed resource discovery on PlanetLab with SWORD. In *First Workshop on Real, Large Distributed Systems (WORLDS '04)*, Dec. 2004.

[23] Public Data Sets on AWS. `http://aws.amazon.com/publicdatasets`.

[24] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy. An Analysis of Internet Content Delivery Systems. In *OSDI*, Dec. 2002.

[25] J. D. Sonnek, A. Chandra, and J. B. Weissman. Adaptive Reputation-Based Scheduling on Unreliable Distributed Infrastructures. *IEEE Transactions on Parallel and Distributed Systems*, 18(11), Nov. 2007.

[26] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of SIGCOMM*, 2001.

[27] R. Trivedi, A. Chandra, and J. Weissman. Heterogeneity-aware workload distribution in donation-based grids. *International Journal of High Performance Computing Applications*, 20(4):455–466, 2006.

[28] R. Wolski, J. Plank, J. Brevik, and T. Bryan. Analyzing Market-Based Resource Allocation Strategies for the Computational Grid. *International Journal of High Performance Computing Applications*, 15(3):258–281, Aug. 2001.