

Figure 5. IaaS cloud infrastructure utilization without backfill VMs, running only on-demand user VMs.

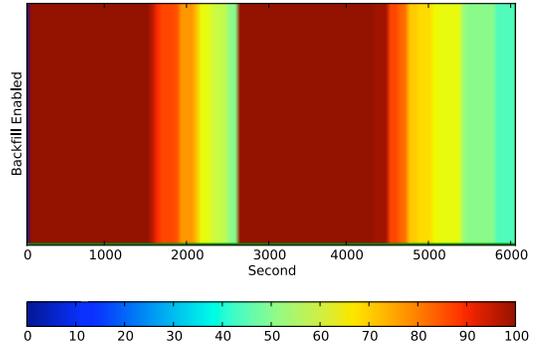


Figure 6. IaaS cloud infrastructure utilization with on-demand user VMs and backfill VMs processing Condor jobs.

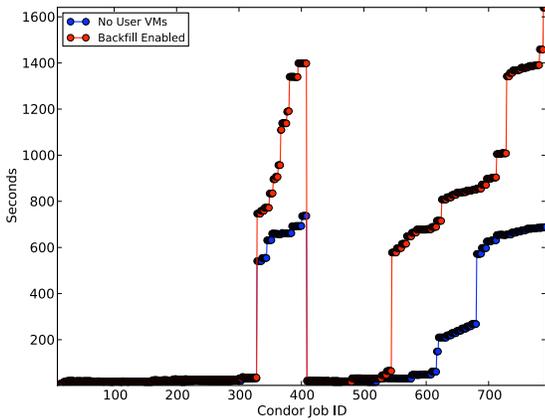


Figure 7. Condor job queued time when the job first begins executing, using the most recent backfill termination policy.

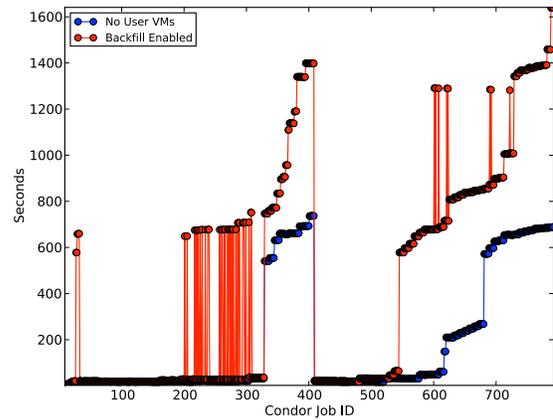


Figure 8. Condor job queued time when the job begins executing for the last time before successful completion, using the most recent backfill termination policy.

we selected this trace because it demonstrates the expected behavior of an overprovisioned cloud infrastructure that is the focus of this work; specifically, it contains many idle VMM nodes available to service on-demand requests.

We emphasize that although we could have generated various workload scenarios, we instead chose these two realistic workload traces in order to demonstrate and evaluate the usefulness of backfill to either the on-demand user community or the HTC user community. (We selected an on-demand trace from the considerably smaller UC Nimbus science cloud than a larger and possibly more dynamic cloud provider, such as Amazon or the Magellan cloud at Argonne National Laboratory [13], because of the lack of availability of such traces at the time of this work.)

Because the UC Nimbus cloud contains only 16 cores and our evaluation environment contains 128 cores, we multiplied the workloads by 8 so that 16 individual requests for the cloud would be 128 individual requests for the entire 128 cores in the evaluation environment. Thus, an individual request for a single core on the UC cloud is 8 individual requests, each for a

single core, in our evaluation environment. The on-demand user workload requests a total of 56 individual VMs over the course of the evaluation. We terminate the evaluation shortly after the overlapping Condor trace completes.

Both workloads submit individual and independent requests; each request is for a single core. In the Condor workload the jobs simply consist of a program that sleeps for the desired amount of time. In the on-demand workload VMs are started and run for the appropriate duration. Backfill VMs are capable of executing 8 jobs concurrently across the 8 cores in a backfill VM, while individual on-demand user requests are single-core VMs. RAM is divided evenly among the VMs.

A. Understanding System Behavior

To understand the system behavior we compare three scenarios. The first scenario considers only the on-demand user workload; the number of cores used in this workload is shown in Figure 2. In this case the IaaS cloud achieves an average utilization of 36.36%, shown in Figure 5, with a minimum utilization of 0% and a maximum utilization of 43.75%.

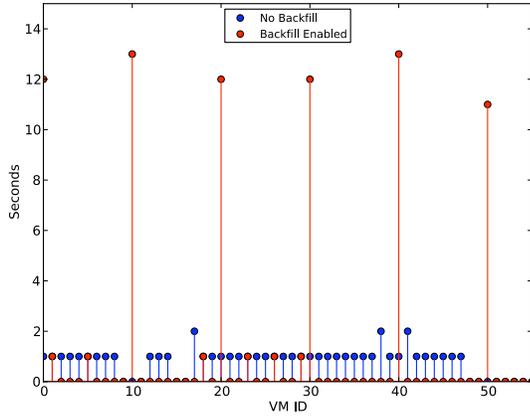


Figure 9. Nimbus service response time for on-demand user VM requests. This is the time from when the Nimbus service first receives the request until the service responds. It includes the time required to terminate backfill VMs (when applicable); however, it does not include the time for a user VM to boot.

The second scenario involves simply running the Condor workload on all 16 VMMs (128 cores) without the on-demand user workload. In this case the entire Condor workload completes in approximately 84 minutes (5042 seconds), as shown in Figure 3.

In the third scenario the Condor workload is overlaid with the on-demand user workload. The Condor workload takes 11 minutes and 45 seconds more than the case where Condor has exclusive access to the resources, completing in approximately 96 minutes (5747 seconds), as shown in Figure 4. However, the utilization of the cloud infrastructure, shown in Figure 6, increases to an average utilization of 83.82% with a minimum utilization of 0% and a maximum of 100%. As the Condor jobs complete (just before 6000 seconds in the evaluation), utilization again drops because the IaaS cloud is no longer running Condor jobs in addition to on-demand user VMs.

The large increase in utilization is due to the fact that the cloud infrastructure is no longer solely dedicated to servicing on-demand user VM requests. Instead, the cloud infrastructure is also able to process jobs from a Condor workload without compromising its ability to service on-demand VM requests. The increase in utilization depends on the amount of work in the HTC workload. Naturally, longer and more HTC jobs will translate into higher utilization.

While increased utilization certainly benefits the cloud provider, Figure 4 also demonstrates that it is advantageous to HTC workloads. The workload, which originally takes approximately 85 minutes on the same dedicated hardware (Figure 3), is delayed by only 11 minutes and 45 seconds (completing in just under 96 minutes) when on-demand user VMs are introduced into the system, as shown in Figure 4. Presumably, however, the cost of utilizing backfill nodes would be lower than utilizing dedicated on-demand user VMs since backfill VMs may be reclaimed by the cloud provider without warning.

B. Understanding System Performance

To understand how the IaaS cloud environment and backfill solution impacts on-demand users and HTC users, we again consider the three scenarios. The first scenario involves the on-demand user workload; the second scenario involves Condor jobs running on the 16 VMM nodes without interruption from on-demand user VMs; and the third scenario overlays the first two.

In Figure 7 we can see that the Condor first queued time is smallest when no user VMs are present, that is, if Condor is allowed exclusive access to its own hardware for executing jobs. Enabling backfill and introducing user VMs causes an increase in the Condor first queued time because there are fewer backfill VMs processing Condor jobs since on-demand user VMs are also running.

When backfill is enabled, there is a noticeable increase in the amount of time that Condor jobs are delayed until they finally begin executing before successful completion, as seen by the numerous spikes for individual Condor jobs in Figure 8 (of which there are a total of 48). These 48 jobs actually first begin executing much earlier, as seen by the absence of spikes in Figure 7. These jobs are delayed because of the arrival of the on-demand VMs, which cause the termination of backfill VMs, preempting the running Condor jobs. Of the 48 jobs that are preempted, the average amount of additional time these jobs are delayed (before they begin executing for the final time) is 627 seconds, with a standard deviation of 76.78 seconds; the minimum amount of extra time that a job is delayed is 273 seconds, and the maximum is 714 seconds. The 48 preempted jobs spent a total of 22,716 CPU-seconds processing the Condor workload before they were preempted. The entire Condor workload required a total of 355,245 CPU-seconds. Thus, for our experimental traces, the use of a backfill-enabled IaaS cloud resulted in an additional 6.39% of overhead for the Condor workload.

Figure 9 demonstrates the impact that backfill has on on-demand user requests. When backfill is disabled, all on-demand user requests are handled in 2 seconds or less. When backfill is enabled, however, the amount of time to respond to an on-demand user request can be as high as 13 seconds, although the majority more closely match the case where backfill is disabled. The large delay in response time occurs when the Nimbus service must terminate (via clean shutdown) backfill VMs in order to service the on-demand user request. Additionally, because the evaluation environment consists of 8-core nodes with backfill VMs consuming all 8 cores, whenever a backfill VM is terminated to free space for an on-demand user VM (even if the on-demand user request is only for a single core), the remaining cores on the VMM node remain idle and freely available for future on-demand user VMs.

While this evaluation is based on two real workload traces, one can imagine that under some of the possible workloads, backfill VMs may be more or less beneficial to IaaS cloud providers and HTC users. Certain workloads, environment characteristics, and backfill termination policies will undoubtedly lend themselves as more beneficial to one community over the other. These are factors that we will consider in future work. However, our backfill solution and

evaluation demonstrate that when considering a realistic on-demand user workload trace and a realistic Condor workload trace, a shared infrastructure between IaaS cloud providers and an HTC job management system can be highly beneficial both to the IaaS cloud provider and to the HTC users by increasing the utilization of the cloud infrastructure (thereby decreasing the overall cost) and contributing cycles that would otherwise be idle to processing HTC jobs.

II. RELATED WORK

Although our work uses backfill to achieve high utilization of an IaaS infrastructure, it is different from work that uses backfill scheduling to increase the utilization of large supercomputers [7]. Scheduling on supercomputers does not typically assume that backfill jobs will be preempted by an on-demand request, seeking to immediately access the resources, while our work assumes this to be the default case. Instead, these backfill scheduling algorithms attempt to backfill unused resources only with requests that match the available slots both in their resource needs as well as their expected runtime. There are, however, preemption-based backfill solutions [22] that share many similar characteristics to our work. The major exception is their focus on queue-based supercomputers and our focus on IaaS cloud infrastructures.

Volunteer computing systems, such as BOINC [2], harvest cycles from idle systems distributed across the Internet. Major examples of volunteer applications include SETI@Home [2] and Folding@Home [11]. These applications are designed to accommodate interruptions in service since widely distributed computers, operated by a seemingly infinite number of disparate users, cannot provide any guarantee of service. In the case of volunteer computing systems, interruptions in service are usually the result of users returning to their systems to do work, systems crashing, or systems becoming disconnected from the Internet. Much research on volunteer computing focuses on the usefulness, efficiency, and failure prediction of these volatile environments [1], [3], [18], [19]. Our work focuses on providing cycles within an IaaS infrastructure that would have otherwise been idle to other processes, such as HTC or volunteer computing, where the services may be interrupted by the arrival of requests for on-demand VMs. Applications that leverage volunteer computing systems would be ideal candidates for backfill VMs because of their ability to handle unexpected failures in service.

In [23] we also leverage recovery techniques, specifically suspending and resuming VMs, to achieve high utilization of IaaS cloud infrastructures. While the goal of maintaining high utilization via introducing different types of leases is the same as the work described here, the leases themselves, as well as the recovery technique used—specifically that of suspending and resuming VMs—are different from the focus in our work. Instead of using suspend/resume to support advanced reservations, we leverage a recovery system that uses resubmission (Condor) to ensure that high utilization is achieved and no work is lost.

Another area that shares related themes to our work is spot pricing, as exemplified by Amazon [2]. With spot pricing, users place bids for instances, and the cloud provider periodically

adjusts the price of spot instances, terminating the spot instances with bids that fall below the new spot price and launching instances that meet or exceed the spot price. Our work uses the current demand for on-demand user VMs to determine the availability for backfill VMs, whereas Amazon bases availability of spot instances on a spot price.

III. FUTURE WORK

The backfill implementation used in this paper was an initial prototype created to demonstrate of the usefulness of combining IaaS cloud infrastructure resources with other purposes, such as HTC, through backfill VMs. The prototype implementation used in this work is publicly available on GitHub [15]. The 2.7 release of the Nimbus toolkit [16] includes the official release of the backfill implementation. In the 2.7 release backfill instances are essentially zero-cost spot instances that have a lower priority than on-demand instances and spot instances. Therefore, backfill instances are preemptible by both on-demand requests and spot requests.

Future work includes exploring different variants of the policies described in Section II, for instance, exploring finer granularity with which to deploy VMs, optimizing the backfill image deployment method, and investigating other termination policies. Another possible area for future work is suspending backfill VMs instead of terminating them. Such a solution may be ideal for a backfill application that does not leverage resubmission as its recovery mechanism.

Another set of challenges arises if we broaden the definition of the preemptible lease, for example, by removing the assumption that only one type of backfill VMs may be used or that only the administrator can configure backfill VMs. One simple refinement would be for the administrator to define multiple backfill VMs and have policies on how backfill resources are shared among them (e.g., what percentage of available cycles should be devoted to each). However, if users are to submit backfill VMs (i.e., the preemptible lease as defined in this paper would no longer be “fixed”), an arbitration mechanism needs to be defined for deciding between various user/instance requests. AWS uses auctions to make such decisions (i.e., spot instances) but many other mechanisms could also be explored. Additionally, we could consider different types of leases, for example, to provide for the impact of backfill VMs on parallel jobs where all processes for a single parallel job must be available.

Other areas for future research involve resource utilization, energy savings, cost, and pricing. An assumption throughout this paper has been that improving utilization is advantageous because it leads to better resource amortization and thus lower costs per computation cycle. This need not necessarily be so, however. Green computing techniques allowing providers to power down a proportion of resources [12] may be a better option in some cases, and prices obtained by auction need not necessarily be sufficient to amortize cost. A more thorough model taking into accounts these factors would be beneficial.

IV. CONCLUSIONS

In this paper we propose a cloud infrastructure that combines on-demand allocation of resources with opportunistic

provisioning of cycles from idle cloud nodes to other processes, such as HTC, by deploying backfill VMs. We extend the open source Nimbus IaaS toolkit to deploy backfill VMs on idle cloud nodes.

We evaluate the backfill solution using an on-demand user workload and an HTC workload. We find backfill VMs contribute to an increase of the utilization of the IaaS cloud infrastructure from 37.5% to 100% during a portion of the evaluation trace but result in only 6.39% additional overhead for processing the HTC workload. Additionally, backfill VMs make available cycles that would have otherwise been idle to assist in processing HTC jobs. In particular, a Condor workload that originally completes in approximately 85 minutes on dedicated hardware is delayed by only 11 minutes and 45 seconds (completing in just under 96 minutes) when on-demand user VMs are introduced into the system.

ACKNOWLEDGMENTS

We thank David LaBissoniere for his help and advice in deploying and evaluating our system on FutureGrid.

This material is based on work supported in part by the U.S. Department of Energy under Contract DE-AC02-06CH11357, in part by NSF SDCI Grant No. 0721867, and in part by NSF Grant No. 0910812 to Indiana University for "FutureGrid: An Experimental, High-Performance Grid Test-bed." Partners in the FutureGrid project include U. Chicago, U. Florida, San Diego Supercomputer Center - UC San Diego, U. Southern California, U. Texas at Austin, U. Tennessee at Knoxville, U. of Virginia, Purdue I., and T-U. Dresden.

REFERENCES

- [1] Acharya A, Edjlali G, Saltz J. "The Utility of Exploiting Idle Workstations for Parallel Computation," SIGMETRICS '97, pp. 225–234.
- [2] Amazon Web Services. Amazon.com, Inc. [Online]. Retrieved December 6, 2010, from: <http://www.amazon.com/aws/>.
- [3] Anderson D, Fedak G. "The Computational and Storage Potential of Volunteer Computing," CCGRID'06, 2006, pp. 73–80.
- [4] Anderson DP, Cobb J, Korpela E, Lebofsky M, Werthimer D. "SETI@home: An Experiment in Public-Resource Computing," Communications of the ACM, 45(11), November 2002, 56–61.
- [5] Anderson, D. "BOINC: A System for Public-Resource Computing and Storage," 5th IEEE/ACM Workshop on Grid Computing, Nov. 2004.
- [6] Thain D, Cieslak D, Chawla N. "Condor Log Analyzer," <http://condorlog.cse.nd.edu>, 2009.
- [7] Feitelson DG, Rudolph L. "Parallel Job Scheduling: Issues and Approaches." In Lecture Notes in Computer Science: Job Scheduling Strategies for Parallel Processing, vol. 949, Springer, Berlin, 1995.
- [8] FutureGrid. [Online]. Retrieved December 6, 2010, from: <http://futuregrid.org/>.
- [9] Internet Retailer Magazine. [Online]. Retrieved December 6, 2010, from: <http://www.internetretailer.com/top500/list/>.

- [10] Keahey K, Foster I, Freeman T, Zhang X. "Virtual Workspaces: Achieving Quality of Service and Quality of Life in the Grid," Scientific Programming Journal, 13(4), 2005, 265–276. (Special Issue: Dynamic Grids and Worldwide Computing).
- [11] Larson SM, Snow CD, Shirts M, Pande VS. Folding@Home and Genome@Home: Using Distributed Computing to Tackle Previously Intractable Problems in Computational Biology. Computational Genomics, Horizon Press, 2002.
- [12] Lefèvre L, Orgerie, AC. "Designing and Evaluating an Energy Efficient Cloud," The Journal of Supercomputing, 51, 2010, 352–373.
- [13] Magellan. [Online]. Retrieved February 11, 2011, from: <http://magellan.alcf.anl.gov/>.
- [14] Michael A, et al., "Above the Clouds: A Berkeley View of Cloud Computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, Feb. 2009.
- [15] Nimbus 2.6 Backfill (prototype). GitHub. [Online]. Retrieved December 6, 2010, from: <https://github.com/pdmars/nimbus/tree/backfill-2.6>.
- [16] Nimbus. [Online]. Retrieved December 6, 2010, from: <http://www.nimbusproject.org/>.
- [17] Nurmi D, Wolski R, Grzegorzczak C, Obertelli G, Soman S, Youseff L, Zagorodnov D. "The Eucalyptus Open-Source Cloud-Computing System.," CCA08: Cloud Computing and Its Applications, 2008.
- [18] Ren X, Lee S, Eigenmann R, and Bagchi S. "Resource Failure Prediction in Fine-Grained Cycle Sharing System," HPDC '06, 2006.
- [19] Ryu K and Hollingsworth J. "Unobtrusiveness and Efficiency in Idle Cycle Stealing for PC Grids," in Proceedings of IPDPS'04, 2004, p. 62a.
- [20] Science Clouds. [Online]. Retrieved December 6, 2010, from: <http://www.scienceclouds.org/>
- [21] Smith JE, Nair R. Virtual Machines: Versatile Platforms for Systems and Processes. Morgan Kaufmann Publishers, San Francisco, 2005.
- [22] Snell Q, Clement M, Jackson D. "Preemption Based Backfill," pp. 24–37 in Job Scheduling Strategies for Parallel Processing, Lect. Notes in Comput. Sci. vol. 2537, edited by DG Feitelson, L Rudolph, and U Schwiegelshohn, Springer Verlag, Berlin, 2002.
- [23] Sotomayor B, Keahey K, Foster I. "Combining Batch Execution and Leasing Using Virtual Machines," 17th International ACM Symposium on High-Performance Parallel and Distributed Computing (HPDC) Boston, June 2008.
- [24] Tannenbaum T, Wright D, Miller K, Livny M. "Condor – A Distributed Job Scheduler," in Beowulf Cluster Computing with Linux, edited by T. Sterling, The MIT Press, 2002.
- [25] Woitaszek M, Tufo H. "Developing a Cloud Computing Charging Model for High-Performance Computing Resources," in IEEE 10th International Conference on Computer and Information Technology, July 2010, pp. 210–217.

The submitted manuscript has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory ("Argonne"). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.