EXFed: Efficient Cross-Federation with Availability SLAs on Preemptible IaaS Instances

Alexander Pucher, Rich Wolski, and Chandra Krintz
Department of Computer Science
University of California, Santa Barbara
{pucher, rich, ckrintz}@cs.ucsb.edu

***Abstract*—Private IaaS clouds offer the benefits of cloud computing on-site but their efficiency is limited by capacity constraints during peak times. We present EXFed, an efficient cross-federation system for IaaS clouds that "ships" jobs between clouds and provides ahead-of-time certainty about resource availability *despite* retaining individual clouds' ability to preempt foreign workload after admission. Clouds participating in the federation remain in control of their local resources at all times and exclusively use a predictable tier of preemptible instances to run federated jobs. This predictable tier is enabled through a new method that provides an SLA on the preemption probability of groups of instances. The SLA is learned statistically from cloud utilization and data transfer rates in the recent past. We deploy EXFed across multiple data centers and evaluate its robustness under realistic and adverse scenarios with production traces recorded from industrial "big data" clouds.**

I. Introduction

The broad availability of open-source Infrastructure-as-a-service (IaaS) cloud frameworks has changed how private Information Technology (IT) infrastructure is built and managed. Private cloud technologies such as OpenStack [1], Eucalyptus [2] and CloudStack [3] enable companies and institutions to deploy IaaS clouds on-site to store data and provision compute resources on-the-fly for a wide range of applications such as web services and "big data" analytics. Similar to public clouds such as Amazon's AWS [4] and Google Cloud Platform [5], private IaaS clouds provide access to their resources under the terms of Service-Level-Agreements (SLAs) that make service quality "guarantees" to resource users. Because the cloud abstraction intentionally obscures the underlying resources from the user (IaaS machines, network, and storage are "virtual"), users must reason about resources in a cloud (public or private) using the terms of the SLA.

Private cloud deployments are often used by organizations with security, regulatory, or cost constraints that make their public cloud alternatives inappropriate. Because the resource pool available in a private cloud is often significantly smaller than in a public cloud, some of the scaling benefits that characterize cloud computing may be lost – in particular for big data workloads generated by frameworks such as Apache Hadoop [6] or Apache Spark [7]. Moreover, this resource scarcity may be intermittent. Thus private cloud operators often overprovision their clouds to absorb the "worst case" aggregate demand for resources. Further, in large organizations with many operational units, the cloud or clouds may be partitioned so that each suborganization is "in charge" of its own resources (e.g. for budgeting purposes). This overprovisioning each cloud or cloud partition exacerbates the problem and leads to significant capital outlay and low utilization.

One way to address the issue of under-utilization is workload federation. We take inspiration from past work on *federation* and *cycle harvesting* [8] in computational grids and build a system that opportunistically "bursts" requests onto remote idle resources if the local system is out of capacity. Grids and clouds, however, have different usage models – grids provide high throughput for queued batch jobs on a best effort basis, whereas clouds offer immediate response to user requests with extensive upfront guarantees about service quality (SLAs). More specifically, the grid's enforcement of local policy via job preemption *after admission* conflicts with the cloud's need to make guarantees about resource availability *ahead of time*.

In this paper, we present EXFed, an efficient cross-federation system for IaaS clouds that "ships" jobs between clouds, provides certainty about resource availability ahead of time, and retains the individual clouds' ability to preempt federated workload after admission. Specifically, we investigate how we can enable a "predictable" preemptible service class of compute instances by providing ahead-of-time availability guarantees, which serves as the basis for opportunistic workload federation across multiple IaaS clouds.

To make a preemptible service class useful for federation, EXFed provides a performance SLA to its users that

- provides a statistical guarantee on the preemption probability of federated workload, and
- considers the latency associated with moving both computation and data of federated instances to and from the execution site.

Specifically, EXFed provides an upper bound on the preemption ratio (the fraction of preempted instances relative to accepted instance requests) for federated workload with a user-specified execution time bound (lifetime bound). Consequently 1.0 minus this fraction serves as a probabilistic "guarantee" that a user's federated instances will execute for at least the requested lifetime once accepted by the remote cloud. The cloud will not accept a request (i.e. the request will "fail fast") if the minimum lifetime cannot be assured probabilistically. Cloud administrators can set the target fraction (i.e. the maximum probability of a preemption) for the preemptible service class so that users can reason about the use of this service class based on its SLA.

Our results show that EXFed is able to maintain a probabilistic SLA on the preemption ratio of federated instances across a broad spectrum of real-world, computationally intensive and big data workloads, even under adverse conditions and tight capacity constraints. Finally, we show that EXFed scales with additional capacity and adapts to changes in cloud configuration and behavior.

II. Method

To represent the combination of computing and storage requirements in a workload, we define a *job* as a homogeneous group of instances and an associated data set in the cloud's object store (*e.g.* OpenStack Swift [9]). The goal, then, is to *predict* whether *each* job that is launched in the preemptible service class will

- transfer its inputs to the target execution site,
- execute for a lifetime specified with the job, and
- transfer any results back to its originating site

before locally generated work at the target site preempts it. EXFed must ensure that the fraction of incorrect predictions (i.e. the prediction error) is below a threshold set by the cloud administrators for the service class.

We define the preemption ratio of federated jobs as $\frac{P}{A}$, where A is the number of jobs submitted and admitted to a remote cloud for federation, and P is the number of jobs in A that are preempted (terminated) by the remote cloud before completion. Thus, the quantity $1.0 - P$ serves as the "success probability" associated with the execution of a federated jobs. Note that in contrast to existing systems implementing preemption [8], [10], our model allows a user to reason about the how long *each* instance will execute before it can be preempted (with a specific probability estimate). Thus, the user's trade-off between preemptible and non-preemptible service tiers is quantifiable ahead-of-time.

To enable ahead-of-time certainty for federated jobs, we introduce admission control that employs a predictive model for deciding whether to accept a request for a preemptible job (a group of instances and data). Admission control is tasked with accepting or rejecting incoming native and foreign job requests based on available capacity without queuing. Native jobs must be accepted as long as (a) sufficient spare capacity is available or (b) local capacity can be made available by terminating foreign jobs. Federated jobs are admitted only if sufficient spare capacity is available (a) to fit the requested job and (b) to guarantee probabilistically that the remaining spare capacity is sufficient to absorb future native requests without triggering preemption.

We assume that each foreign job requires inputs from the originating cloud's object store and that outputs from the job must be returned to the originating cloud. Further, we assume that there is sufficient storage in the object store of the cloud accepting a federated job to hold that jobs' inputs and outputs temporarily. As a consequence, preemptions are only triggered due to instance capacity constraints, not storage shortfall.

Key to our approach is that federated job requests come with a user-specified *upper bound on the lifetime* for the job's execution duration until completion. A cloud only accepts a federated job (admits a preemptible job) if its upper bound on lifetime is shorter than the lower bound estimate of the job's "time-to-preemption", subject to a confidence level defined in the cloud's SLA. Further, SLAs in our system are defined upon job submission and are immutable for the lifetime of a job.

A. Estimating Native Load Increase

The intuition behind our admission control mechanism is that the job preemption probability depends on the current load level of the cloud as well as predictable changes of the load level in the near future. For example, when the cloud's load is near capacity, new native workload is more likely to cause a preemption of foreign workload than if the cloud is relatively under utilized. Notice that it is only the *arrival* of new native instances (i.e. a load increase) that can trigger the termination of preemptible instances – foreign instances will simply be rejected by admission control if there is insufficient spare capacity to host them.

When the admission control algorithm considers a new federated job request, it requires a prediction of the time until spare capacity is exhausted (and at least one additional native instance arrives, triggering preemption). We refer to this time estimate as the "time-to-preemption".

However, for responsiveness and scalability reasons, rather than making this estimate on a per-request basis, EXFed continuously computes the time until there will be a capacity shortfall for different possible federated job sizes. To do so, we sample the history of total capacity utilization of native instances at regular intervals. From each sample, we trace forward to identify the point at which the aggregate utilization of non-preemptible instances (native instance starts without compensating native instance terminations) increases by a fixed-size step (e.g. one instance slot). We repeat this tracing procedure for each possible magnitude of load change (two slots, three slots, and so on) and tabulate the results as a set of empirical distributions. This database of predictions is constantly updated but queried asynchronously by the admission control component. When EXFed considers a new request for a federated job, it computes the time-to-preemption by retrieving the distribution corresponding to the load increase that is equal to the current level of available spare capacity (including the new job) plus one (the hypothetical native instance triggering preemption) from the table and uses a quantile corresponding to the SLA from this distribution as the estimate.

B. Admission Control

For every admission decision, the admission control component runs a discrete-event simulation in which there are two types of events: the termination of a foreign job, and the increase in occupancy of the cloud by native workload. It generates a prediction of the available spare capacity over time taking into account the remaining lifetimes of the current foreign job set and makes decisions based on whether new requested foreign job will "fit" within the available capacity. Note, that in addition to lifetimes, EXFed also estimates overheads for transmitting data between federating clouds to determine whether a job can execute in time.

When EXFed considers admitting a new foreign job, it first computes the number of "slots" – units of cloud allocation –

that are occupied by the current set of foreign jobs running in the cloud, and the number of slots that are occupied by native workload. If there are enough free slots to host the new foreign job, it then, hypothetically, adds the job to the set of foreign jobs in the cloud (otherwise the job is rejected).

We next predict a lower bound on the time until the number of unused slots will be consumed by native work with a probability of 1.0 minus the SLA preemption probability (e.g. 0.95 for a preemption probability of 0.05). If the new foreign job's duration (i.e. its lifetime plus expected startup and teardown overheads) is greater than or equal to this bound, the job is rejected.

Otherwise our system sorts the foreign jobs by their termination times (which were computed from upper bounds when the jobs were admitted) and "rolls" time forward from one job completion to the next. Each time a foreign job terminates the admission control component again predicts the bound on the time until native workload will exhaust available slot capacity. If the remaining duration of the new job, at each of these termination points, is greater than the prediction, the new job is rejected. If there is no point in time at which the simulation predicts that native workload will exhaust capacity (and thereby cause a preemption) between the time the new foreign job is submitted and the end of its duration, the new foreign job is admitted.

III. Results

Our evaluation attempts to answer three primary questions about EXFed's new approach to cloud federation, which provides an ahead-of-time guarantee on the preemption probability of federated jobs executed on preemptible resources:

- Is it feasible to enforce an upper bound SLA on the preemption ratio of federated jobs with real-world workloads?
- Does the method scale with additional capacity to a federation of clouds while maintaining its guarantees?
- How sensitive is the method to properties of real-world workloads and adverse degrees of resource contention?

A. Experimental Methodology

To answer the three questions, we emulate various federation settings via faster-than-realtime and smaller-than-production replay of workloads recorded from production clouds, using real-world private clouds equipped with our federation extensions. We choose this empirical approach over a simulation approach so that the experiments take into account the overheads (modeled and unmodeled) of working clouds. That is, while the workload is a replay of production workloads (sped up from their original durations), the clouds are real. The experiments transfer data across a wide area network between federating clouds, launch and terminate instances, and retrieve data representing generated results.

We install Eucalyptus 4.1 from repository packages on CentOS 6.7. Due to the long duration of the experiments we use several different clusters located at CloudLab [11] APT Utah, CloudLab Clemson, and UC Santa Barbara with 4-8 physical hosts each.

TABLE I: Production traces. Types are batch (B), batch pipelined (P), batch seasonal (S), long-running service (L), and mixed (M).

segment	num reqs	num insts	total data (TB)	trace dur (wks)	average inst dur (hours)	work type
DS-A 1	2.3k	3.0k	1100	2	2.2	B, S
DS-A 2	1.5k	2.0k	50	2	2.9	B, S
DS-B 1	1.3k	4.2k	80	12	2.2	B, P
DS-B 2	1.5k	2.4k	180	12	9.3	M
DS-C 1	1.2k	2.8k	–	30	10.0	M
DS-C 2	0.5k	1.9k	–	30	45.5	B, L

B. Production Trace Data Sets

Our evaluation uses three groups of anonymized, real-world traces recorded from production systems, which we obtained from industry collaborators (Table I). Each represents two separate sub-traces with jobs recorded from "big data" clusters. They span multiple months in real time and contain mixed workloads from multiple cloud users and software frameworks. For our replay we extract long-running jobs over 1 hour in real-time duration, scale them to fit our test bed, speed up replay by a factor of $50x$, and use separate parts of the traces for model training and evaluation.

During our development, DS-C was available for testing and should be considered in-sample (train-train). Part of DS-A was available during development as well, although the evaluation uses a more recent trace segment that was unavailable during our development. DS-B became available only after we completed the development of our prototype and thus represents a true out-of-sample test (train-test).

C. Evaluation Metrics

We consider three primary metrics to evaluate EXFed: the preemption ratio of federated jobs, the rejection ratio of requests, and the federation ratio of work. Preemption ratio measures the effectiveness of SLA enforcement:

$$preemption = \frac{|federated\_jobs\_preempted|}{|federated\_jobs\_accepted|}$$

This metric captures the fraction of pre-mature terminations of federated jobs (groups of instances) that have been accepted by the admission controller and launched in the system.

Rejection ratio measures the efficacy of federation in accepting incoming user requests:

$$rejection = \frac{|jobs\_rejected|}{|jobs\_requested|}$$

A rejected request has been rejected by the local cloud due to a lack of capacity and by the remote clouds in the federation due to the inability to guarantee the requested SLA preemption ratio. We only count jobs that are rejected first as native requests and subsequently as federated requests once.

The federation ratio represents the overall mutual gain from federation and measures the amount of work that results

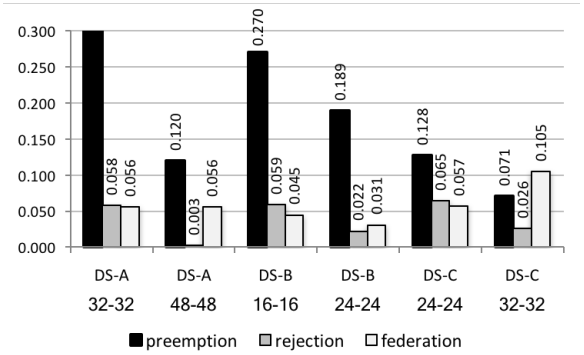

Fig. 1: Federating workloads between two capacity-constrained clouds without SLA enforcement for 3 production traces for clouds with different capacities. Federation enables more jobs at the cost of more preemptions.

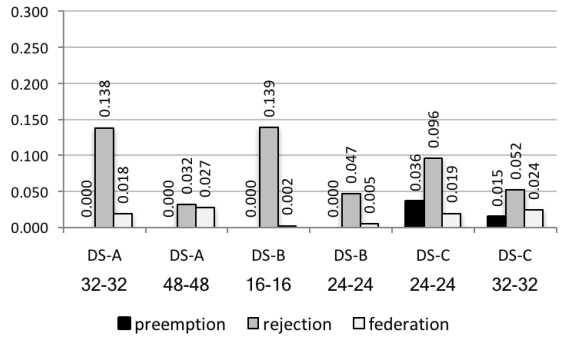

Fig. 2: Federating workloads between two capacity-constrained clouds with SLA enforcement enabled. Admission control rejects some jobs to guarantee a 0.05 upper bound on the preemption ratio.

from access to remote capacity (completed jobs on remote, preemptible instances).

$$federation = \frac{work(federated\_jobs\_completed)}{work(jobs\_completed)}$$

The ratio uses the aggregate time spent by instances of successfully completed jobs. It divides the aggregate time of completed federated jobs by the aggregate time of all completed jobs (including federated jobs). This time includes the amount of time spent to set up instances, wait for data transfers, execute the actual computation tasks, and tear down the instances. Time spent by preempted jobs is not included.

D. Federation Baseline (no SLA)

Figure 1 shows six *baseline* experiments, replaying our three production data sets on two different, real cloud configurations of EXFed each – without SLA-enabled admission control (i.e. without prediction or capacity planning). In these experiments, admission of federated jobs is based purely on available spare capacity (i.e., all foreign job requests are accepted if, at the time of their arrival, there is sufficient capacity to host them). Each group of columns on the x-axis represents one experiment, executing the sub-traces of a data set concurrently on two clouds linked via two-way federation. The category names describe the experimental setup for each data set, e.g. "DS-A", and the size of the clouds in the federation, e.g. "32-32" which indicates that each cloud employs 32 cores each.

We replay the three production data sets on two different cloud configurations. The first configuration tightly constrains the amount of available resources to show the federation's behavior under adverse conditions. The second configuration provides additional capacity as in a an overprovisioned (cost-inefficient) setting. On the y-axis we plot our three performance metrics: the preemption ratio (red), the rejection ratio of requests overall (blue), and the ratio of successfully federated (additional completed) work (green).

The figure shows how the relationship between workload and capacity impact each metric. The degree to which additional capacity affects EXFed's baseline federation activity and the preemption ratio depends on the workload. DS-A has a similar federated work ratio in both configurations. As such, extra capacity is devoted relatively equally to federated and native jobs. For DS-B, extra capacity hosts more native workload (the rejection ratio and the federation ratio both decrease). For DS-C, the added capacity is used primarily to host federated workload (the rejection ratio decreases but the federation ratio increases).

E. Federation With SLA Guarantees

Figure 2 shows the results for the same setup on real clouds as the previous section, but with EXFed's admission control actively enforcing the preemption SLA. Our method reduces the number of jobs admitted in order to meet its preemption ratio guarantees (using the 0.05 upper bound). The results show that EXFed maintains a preemption ratio below 0.05 for all experiments indicating that ahead-of-time guarantees on the preemption ratio of federated instances are possible. This result holds for differing levels of capacity constraints across production traces.

Compared to the baseline results (Figure 1), EXFed's admission control provides certainty about preemption probability but increases (federation) rejection ratios. The native work (not shown) is unaffected, but the work performed by federated instances decreases. That is, our method trades off a small amount of the total federated work completed to provide an SLA on preemption ratio for federated jobs. These results indicate that our method changes the way additional capacity is used for federation. Because admission decisions depend on the absolute amount of available spare capacity at the time of the request, EXFed is less sensitive to workload properties and achieves higher federation ratios on larger clouds.

We also observe that under SLA constraints, DS-B federates few jobs compared to DS-A and DS-C. The reason for this is that for both sub-traces (one per cloud in the federation), DS-A and DS-C contain large numbers of small and short-running jobs, with DS-C also containing a few long-running, service-like requests in sub-trace two. DS-B is different in that sub-trace one contains primarily short jobs with a large number of instances per request and sub-trace two contains many long jobs with a small number of instances per request. This mismatch in workload characteristics between the two

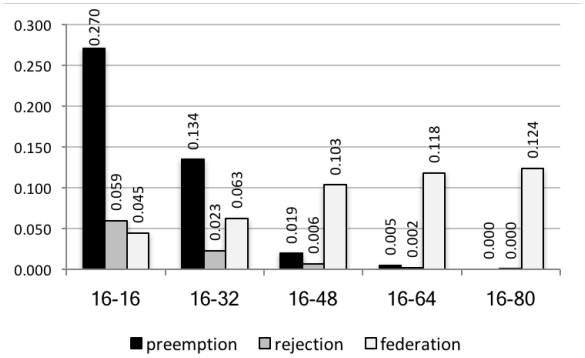

Fig. 3: DS-B on multiple cloud configurations without SLA enforcement. The preemption and rejection ratios decrease as capacity is added to the cloud federation (left to right) while the federation ratio increases.

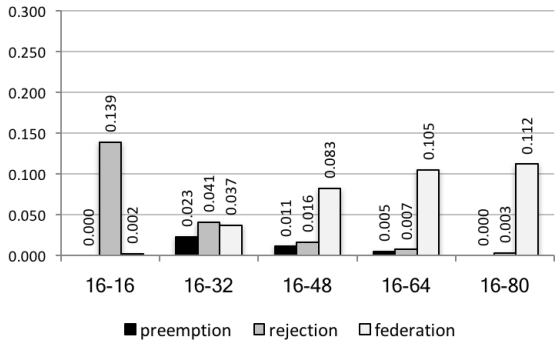

Fig. 4: DS-B on multiple cloud configurations with SLA enforcement enabled. The preemption ratio remains below the 0.05 upper bound while the federation ratio increases with additional capacity.

clouds in the federation, makes it more challenging to federate jobs in DS-B compared to DS-A and DS-C.

F. Scaling With Cloud Capacity

We next evaluate EXFed's ability to scale with added cloud capacity. Specifically, we keep the workload and number of clouds within the federation constant and vary the total capacity of a single cloud. When resource constraints are removed by adding sufficient capacity accommodate all requests at any given time, SLA enforcement becomes unnecessary and may effectively reduce utilization by imposing overhead. If our method scales gracefully with capacity, it will admit additional federation requests as the total capacity of the participating clouds increases. Asymptotically, with increased capacity, our method should complete a similar amount of work as the no-guarantees baseline – with the added benefit of ahead-of-time certainty about preemption probabilities. In particular, we investigate the capacity scaling behavior of our admission control for DS-B. This data set is of special interest as DS-B represents our out-of-sample test and appears ill-suited for federation when highly resource constrained.

Figures 3 and 4 show the results of our capacity scaling experiments for DS-B for the baseline and SLA-enabled admission control, respectively. The categories on the x-axis again describe the size of the clouds in the federation. The left figure in each pair shows federation statistics without SLA-enforcement, the right with SLA-enforcement enabled with an upper bound 0.05 preemption ratio of federated instances. Federation capacity increases from left to right in each graph. While the size of the cloud running sub-trace 1 is fixed at 16cores, the size of the cloud running sub-trace 2 increases in steps of 16 cores, for a total federation capacity of 80 cores.

Figure 3 represents the baseline for scaling cloud capacity with DS-B. The smallest configuration "16-16" results in a 0.27 preemption ratio. As the second cloud in the federation increases in size, the ratios decrease. Concurrently, the federation ratio increases from 4% to 12%. With a manual overprovisioning approach to SLA-enforcement we observe an optimal capacity of cloud 2 to be approximately 48 cores.

Figure 4 shows results of using SLA-enabled admission control for DS-B. The smallest configuration "16-16" results in a preemption ratio near 0 because very few jobs can be federated. As the federation capacity increases, the rejection ratio decreases while the federation ratio increases. Moreover, EXFed maintains a preemption ratio below 0.05 for all configurations, remaining in the 0.01 to 0.02 range.

In results omitted for brevity the comparison of scaling experiments for DS-A, DS-B and DS-C offers an additional insight into the traces. The results indicate that the strongly seasonal DS-A has more potential for federation than DS-B, while DS-C behaves similarly to DS-B with a sharp decline of federation activity under SLA- and capacity constraints.

These results show that EXFed is able to scale gracefully with capacity and enables us to quantify the opportunity cost for automatically enforcing an SLA on the preemption ratio of federated jobs. Our admission control consistently maintains a preemption ratio below its 0.05 target and increases the amount of admitted federated work as capacity is added to the clouds. The differences in work completed between the baseline and the SLA-enabled cases are most visible in resource constrained settings. As capacity is added to the clouds, admission control accepts additional federated jobs, reaching federation ratios within 20% of the baseline.

G. Scaling with Federation Size

The utility of cloud federation capabilities should increase with a larger pool of workload available for consolidation. We investigate whether EXFed delivers increased efficiency with federation size by comparing the performance of different federation setups with four individual clouds. Specifically, we use four clouds executing fixed workload traces for federation setups with 4 unconnected clouds, 2 unconnected and two federated clouds, two separate federations with two clouds each, and a single federation of all four clouds. If federation adds value with increasing federation capacity and size of the user pool, the fully federated setup should perform best while the unconnected setup should perform worst.

Our experiment is similar to the scaling experiments in Section III-F above, although we use DS-A to represent our workload. As cloud configurations, we chose four clouds with a configuration of "32-56-32-56", indicating two clouds with 32 cores each and two clouds with 56 cores each. We refer

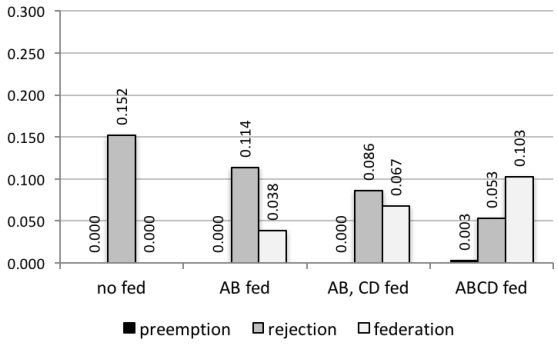

Fig. 5: Efficiency gains for four clouds (A, B, C, D) with increasing federation size from left to right. Baseline (left), two separate clouds and a federation of two clouds, two federations of two clouds each, federation across all four clouds (right). As the number of federated clouds increases, the aggregate rejection ratio decreases while federated work increases.

to these clouds as "A", "B", "C", and "D" respectively. To obtain the 4 different traces necessary to drive all four clouds, we use an new equal-length trace DS-Ax that was recorded immediately following the end of the original DS-A trace. We "time-shift" this new trace to simulate a workload executing concurrently with DS-A.

Figure 5 shows the increasing benefit of adding opportunities for workload consolidation by joining clouds together in a larger federation. We plot four different federation setups as categories on the x-axis and again plot the aggregate preemption ratio, rejection ratio and federated work ratio over all four clouds on the y-axis. The leftmost setup represents the baseline, with all four clouds executing their workloads without federation, while the rightmost setup allows federation across all four clouds. In all setups the clouds are of constant size and replay their specific workload traces.

We observe an incremental improvement in the aggregate amount work accepted by the clouds, as indicated by a decreasing rejection ratio from left to right. Equivalently, as the pool of workload and capacity increases by incrementally joining clouds together by federation, the amount of federated work increases. Admission control performs correctly in all scenarios and maintains a preemption ratio well below the 0.05 threshold. The most relevant result is the improvement from two federations of two clouds each to federation across all four clouds. As the federation grows in total capacity and user base – with constant size and workload – the smoothing of demand leads to greater consolidation benefits.

IV. Related Work

Eucalyptus [2], OpenStack [1], CloudStack [3] and OpenNebula [12] are open-source IaaS cloud frameworks. All four support limited federation capabilities as part of a "hybrid cloud" setup, albeit focused on API-compatibility and user authentication.

The use of preemptible instances and resulting cost savings and uncertainty has been studied in the literature [13], [14], [15], [16]. Andrzejak et al. [17] use AWS spot price time series to construct probabilistic bounds on the execution time of jobs, assuming the presence job check-pointing capabilities.

Grozev and Buyya [18] survey a corpus of work in the "inter-cloud" federation context. Researchers propose many different architectures using brokers [19] and PaaS layers [20], [21]. RESERVOIR [22] is a peer-to-peer federation system between clouds to using job-level granularity, but does not consider the interaction of SLAs and job preemption.

Our system, EXFed uses a de-centralized federation architecture with job-level granularity. In contrast to existing work, our approach allows each member cloud to remain in control of its own resources at all times (i.e. receiving clouds do not extend open-ended availability guarantees to federating clouds), while still providing users with an SLA guarantee on job completion probability. The SLA is available ahead-of-time and is based on a statistical model learned from historic cloud behavior. In our evaluation we investigate that the SLA on job preemption fraction is met consistently for a variety of realistic and adverse workload scenarios, rather than attempt to optimize a monetary "profit" function. We also evaluate our system in a real-world deployment by replaying scaled-down workloads recorded from production systems across multiple data centers.

V. Conclusion

We present EXFed, an efficient cross-federation system for IaaS clouds that "ships" jobs between clouds and provides ahead-of-time certainty about resource availability despite retaining the individual cloud's ability to preempt foreign workload after admission. Clouds participating in the federation remain in control of their local resources at all times by exclusively using a predictable tier of preemptible instances to run federated jobs. This predictable tier is enabled through a new method that provides an SLA on the preemption probability of groups of instances. The SLA considers data migration and computation and derives from cloud utilization traces in the recent past. The SLA guarantee is learned statistically, is specific to each cloud participating in the federation, and adapts to changes in utilization and transfer bandwidth.

We implement EXFed end-to-end on top of the Eucalyptus IaaS framework. We deploy EXFed across multiple data centers and evaluate the robustness of our SLA guarantees under realistic and adverse scenarios with a battery of production traces recorded from industrial "big data" clouds. We find that EXFed consistently maintains the SLA on the preemption probability of federated jobs and is robust to real-world conditions such as seasonal patterns in load and resource contention.

Acknowledgments

We thank our industry collaborators Altiscale and Turn for providing us with anonymized workload traces. We also thank CloudLab for providing capacity for our experiments. This work is funded in part by NSF (CCF-1539586, CNS-1218808, CNS-0905237, ACI-0751315), NIH (1R01EB014877-01), and the California Energy Commission (PON-14-304).

REFERENCES

[1] "OpenStack," [Online; accessed Aug-2014] "http://www.openstack.org/".

[2] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on. IEEE, 2009, pp. 124–131.

[3] "CloudStack," [Online; accessed Aug-2014] "http://cloudstack.apache.org/".

[4] "Amazon Web Services home page," http://aws.amazon.com/.

[5] "Google Cloud Platform," http://cloud.google.com/, [Online; accessed 01-May-2016].

[6] "Hadoop MapReduce," "http://hadoop.apache.org/".

[7] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets." HotCloud, vol. 10, pp. 10–10, 2010.

[8] D. H. Epema, M. Livny, R. van Dantzig, X. Evers, and J. Pruyne, "A worldwide flock of condors: Load sharing among workstation clusters," Future Generation Computer Systems, vol. 12, no. 1, pp. 53–65, 1996.

[9] "OpenStack Swift," http://swift.openstack.org/, [Online; accessed 01-May-2016].

[10] J. Barr, "Amazon EC2 Spot Instances And Now How Much Would You Pay?" Dec. 2009. [Online]. Available: https://aws.amazon.com/blogs/aws/ec2-spot-instances-and-now-how-much-would-you-pay/

[11] R. Ricci, E. Eide, and C. Team, "Introducing cloudlab: Scientific infrastructure for advancing cloud architectures and applications," Login USENIX Magazine, vol. 39, no. 6, December 2014.

[12] D. Milojičić, I. M. Llorente, and R. S. Montero, "Opennebula: A cloud management tool," IEEE Internet Computing, vol. 15, no. 2, pp. 11–14, 2011.

[13] S. Subramanya, T. Guo, P. Sharma, D. Irwin, and P. Shenoy, "Spoton: a batch computing service for the spot market," in Proceedings of the Sixth ACM Symposium on Cloud Computing. ACM, 2015, pp. 329–341.

[14] N. Chohan, C. Castillo, M. Spreitzer, M. Steinder, A. Tantawi, and C. Krintz, "See spot run: Using spot instances for mapreduce workflows," in Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing, ser. HotCloud'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 7–7. [Online]. Available: http://dl.acm.org/citation.cfm?id=1863103.1863110

[15] M. Mazzucco and M. Dumas, "Achieving performance and availability guarantees with spot instances," in High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on, Sept 2011, pp. 296–303.

[16] J. Chen, C. Wang, B. B. Zhou, L. Sun, Y. C. Lee, and A. Y. Zomaya, "Tradeoffs between profit and customer satisfaction for service provisioning in the cloud," in Proceedings of the 20th International Symposium on High Performance Distributed Computing, ser. HPDC '11. New York, NY, USA: ACM, 2011, pp. 229–238. [Online]. Available: http://doi.acm.org/10.1145/1996130.1996161

[17] A. Andrzejak, D. Kondo, and S. Yi, "Decision model for cloud computing under sla constraints," in Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on, Aug 2010, pp. 257–266.

[18] N. Grozev and R. Buyya, "Inter-cloud architectures and application brokering: taxonomy and survey," Software: Practice and Experience, vol. 44, no. 3, pp. 369–390, 2014.

[19] R. Buyya, R. Ranjan, and R. N. Calheiros, "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services," in Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing - Volume Part I, ser. ICA3PP'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 13–31. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-13119-6_2

[20] D. Villegas, N. Bobroff, I. Rodero, J. Delgado, Y. Liu, A. Devarakonda, L. Fong, S. M. Sadjadi, and M. Parashar, "Cloud federation in a layered service model," Journal of Computer and System Sciences, vol. 78, no. 5, pp. 1330–1344, 2012.

[21] C. Bunch and C. Krintz, "Enabling automated hpc/database deployment via the appscale hybrid cloud platform," in Proceedings of the first annual workshop on High performance computing meets databases. ACM, 2011, pp. 13–16.

[22] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres et al., "The reservoir model and architecture for open federated cloud computing," IBM Journal of Research and Development, vol. 53, no. 4, pp. 4–1, 2009.