Yin and Yang: Balancing Cloud Computing and HTC Workloads

Zhuangwei Kang
Vanderbilt University
Nashville, Tennessee
zhuangwei.kang@vanderbilt.edu

Zhuo Zhen
University of Chicago
Chicago, Illinois
zhenz@uchicago.edu

Kate Keahey
Argonne National Laboratory
Chicago, Illinois
keahey@mcs.anl.gov

1 INTRODUCTION

Resources for computer science systems research need to be available on-demand to support interactive exploration; this requires over-provisioning of infrastructure and results in resources under-utilization (Figure 1). High Throughput Computing (HTC) [1] involves running a large number of independent tasks implementing a domain science application. HTC does not require interactivity and is designed to be resilient to resource loss: tasks that do not run to completion are simply re-executed, which may waste time and energy.

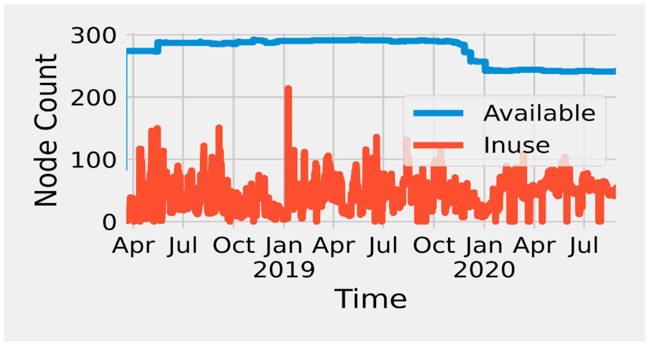

Figure 1: Resource Utilization of ComputeHaswell nodes in Chameleon

These two models – on-demand and on-availability scheduling – are often seen as incompatible leading to "resource silos", i.e., resources that can only be used for this or that model, thus lowering utilization potential and flexibility, as well as increasing costs. This raises a question: can we combine academic clouds like Chameleon[5] and the HTC systems like Open Science Grid (OSG)[9] within one system such that objectives of both are satisfied? Commercial cloud providers use preemptible instances to solve a similar problem. Our objective is to explore a similar approach in academic clouds, via the use of preemptible instances that encapsulate HTC workloads and can be deployed when the system is not needed for interactive use.

Figure 2 shows a snapshot from the Chameleon calendar that shows that there are gaps in resource usage between leases. In this poster, we seek to answer the following questions:

(1) Can we fill these lease gaps by deploying HTC jobs?
(2) What is the most efficient strategy to do this?
(3) How can we minimize the cost of doing this by adopting strategies that reduce HTC reruns?

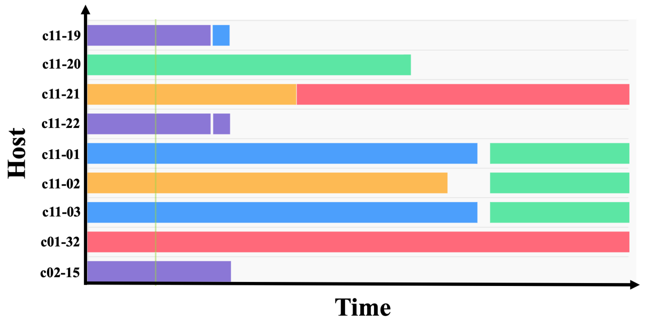

Figure 2: Gantt Chart of Chameleon Host Reservations [2] (Colorful bars represent leases, blank spaces indicate devices are spare during the period.)

2 APPROACH

We developed a simulator CHISim (Figure 3) that replicates the components and processing logic of the OpenStack Blazar[1], the Chameleon resource manager. Similar to the Blazar API, the FrontEnd in CHISim accepts cloud lease requests and communicates with the Resource Manager to assign resources to cloud users. Lease requests are then forwarded to the Request Forecaster which predicts requests for the next time slot. Based on those predictions the HTC job scheduler works with the Resource Manager to assign idle resources to HTC jobs, and also to preempt HTC instances when the nodes are requested by the cloud users.

In this work, we evaluated four preemption policies:

(1) **Random**: preempt nodes from the HTC pool arbitrarily;
(2) **Recent-Deployed**: preempt nodes that are assigned to HTC most recently;
(3) **Least-Core-Used**: preempt nodes with the least number of cores assigned to HTC jobs;
(4) **Least-Resubmit**: preempt nodes with the least number of re-submissions.

The CHISim handles the unfinished jobs by placing them at the head of the HTC queue and re-executing them from the initial state. The Resource Manager sends advance notice of preemption to HTC Scheduler when the Request Forecaster predicts an upcoming takeaway of the resources from the cloud users.

The Resource Forecaster supports three prediction models (Figure 4): **Rolling-Mean**, **Rolling-Median**, and an **Long Short-term Memory (LSTM) -based deep learning model**. The initial LSTM model is trained offline with the Chameleon cloud trace dataset [3] (March 2018 to May 2020), and updated online periodically.

[1] https://docs.openstack.org/blazar/latest/

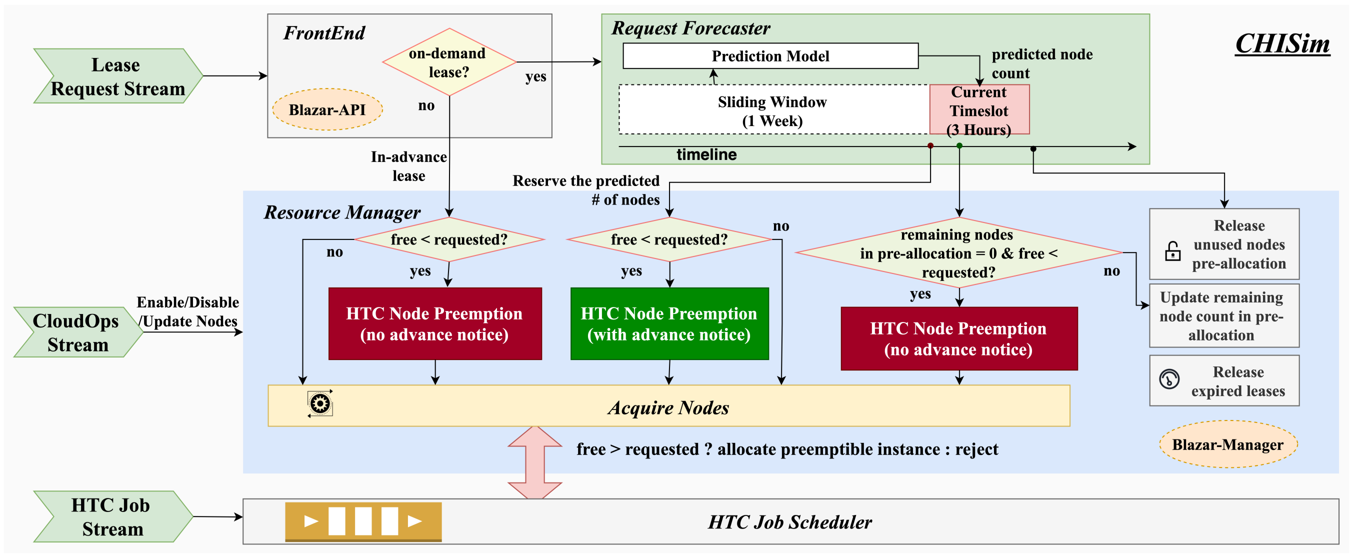
CHISim
Lease Request Stream
FrontEnd
on-demand lease?
yes
no
Blazar-API
In-advance lease
Request Forecaster
Prediction Model
predicted node count
Sliding Window (1 Week)
Current Timeslot (3 Hours)
timeline
Reserve the predicted # of nodes
Resource Manager
free < requested?
no
yes
free < requested?
no
yes
remaining nodes in pre-allocation = 0 & free < requested?
yes
no
HTC Node Preemption (no advance notice)
HTC Node Preemption (with advance notice)
HTC Node Preemption (no advance notice)
Release unused nodes pre-allocation
Update remaining node count in pre-allocation
Release expired leases
CloudOps Stream
Enable/Disable /Update Nodes
Acquire Nodes
free > requested ? allocate preemptible instance : reject
Blazar-Manager
HTC Job Stream
HTC Job Scheduler

Figure 3: CHISim Architecture

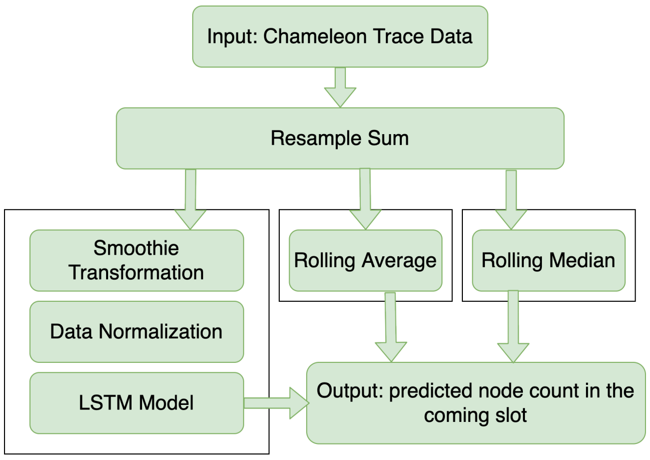

Figure 4: Request Forecasters

3 EVALUATION

With the simulator, we conducted and compared the following experiments on three metrics - resource utilization improvement, energy efficiency, and HTC node preemption.

(1) **Baseline**: run Chameleon user requests only(no advance notice for preemption);
(2) **Greedy algorithm**: filling lease gaps with HTC jobs on all available resources(no advance notice for preemption);
(3) **Predictive filling**: use forecasters and preemption policies (conducted 12 experiments with 4 preemption policies and 3 prediction models).

3.1 Resource Utilization Improvement

In general, the average usage rate increases by 74.4%-75.9% (i.e. figure 5) with the predictive filling methods relative to the baseline.

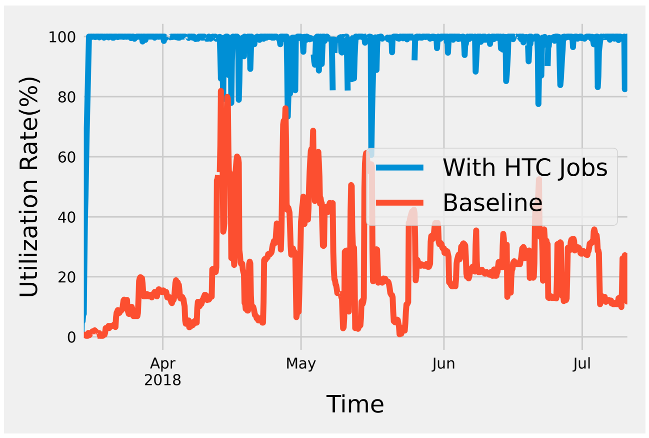

Figure 5: Resource Utilization Rate (read line: Chameleon baseline usage, blue line: the total usage after deploying HTC jobs with the Rolling-Median-based Request Forecaster and the Recent-Deployed preemption policy)

Utl Degradation(%) Policy / Forecaster	Random	Least-Core-Used	Least-Resubmit	Recent-Deployed
Rolling-Mean	0.96	0.84	0.91	0.49
Rolling-Median	0.45	0.21	0.23	0.29
LSTM	1.22	0.99	0.95	1.10

Table 1: The differences between predictive filling approaches and the greedy algorithm in the average resource utilization rate ($Utl_{greedy} - Utl_{predictive}$). Rolling-Median introduces the lowest cost than other methods.

The Request Forecaster reduces the utilization rate compared to the greedy algorithm due to the over-provision caused by prediction errors (Table 1).

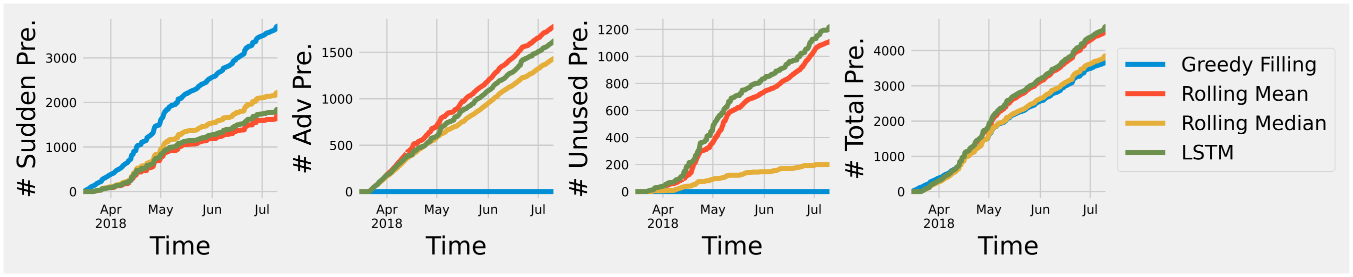

Figure 6: HTC Node Preemption (preemption policy: Recent-Deployed)

3.2 Energy Efficiency

We measured the energy efficiency using the wasted core hours of uncompleted HTC jobs due to preemption. The waste rate is defined in equation 1. The Recent-Deployed policy has the lowest waste rate because there is no wasted core hours if the instance is preempted during the overhead time of re-configuring the node for the HTC jobs.

$$wr = \frac{1}{N}\sum_{n=1}^{N}\frac{100 * CoreHours(wasted, i)}{CoreHours(total, i)} \tag{1}$$

Waste Rate(%) Forecaster / Policy	Greedy Filling	Rolling-Mean	Rolling-Median	LSTM
Random	4.64	5.57	4.16	6.05
Least-Core-Used	3.69	4.59	3.85	5.00
Least-Resubmit	3.88	4.54	4.54	5.00
Recent-Deployed	2.95	2.62	2.09	2.71

Table 2: Mean Energy Waste Rate (energy is measured as core hours; shadowed cell indicates the winner; the Recent-Deployed preemption policy with the Rolling-Median Request Forecaster uses energy more efficiently.)

3.3 HTC Node Preemptions

Node Preemption Category	Description
Sudden	The Forecaster underestimates the Chameleon user requests, and the HTC jobs will be terminated without advance notice.
Unused	The Forecaster overestimates the user requests, leading to unnecessary preemptions.
Advanced	HTC nodes are preempted with advance notice and preempted resources are used by Chameleon users as predicted.

Table 3: Node Preemption Categories

According to the difference between the predicted Chameleon user requests and the true value, HTC node preemptions can be classified into three categories (Table 3). It can be seen from Figure 6 that (1) enabling the Request Forecaster reduces the number of sudden preemptions by 42.15% - 54.9% relative to the greedy algorithm; (2) Rolling-Median-based forecaster produces more sudden but less unused and total preemptions; (3) LSTM and Rolling-Mean yield more unused preemptions.

4 RELATED WORK

Providing preemptible instances can maximize the revenues of the commercial cloud vendors and provide users with cost-effective and partially reliable services. [8] In our work, we borrowed the concept and used the preemptible instances to increase the resource utilization in academic cloud testbeds by offering partially QoS guarantee to the HTC users. Garcia et al. [4] designed a schema for pluggable preemptible-aware VM scheduler, but failed to compare various scheduling policies and to discuss the QoS of the HTC jobs. In [7], the proposed scheduler only considered the submitted HTC jobs. Without predicting the future allocation requests, the scheduler yielded excessive unfinished HTC jobs, which caused the energy-waste. Unlike virtual resources on a regular on-demand cloud, physical resources on Chameleon must be reserved before using them for an experiment. Once a reservation has been accepted, users are guaranteed that resources will be available at the time they chose, which helps to plan large scale experiments. The uniqueness of Chameleon requires a modified solution to our previously proposed infrastructures [6] to reconcile with the HTC systems.

5 CONCLUSION

Based on the observations, we provide the following take-home messages on choosing the request forecasters and preemption policies:

(1) Combining Chameleon and HTC workloads can increase utilization without compromising the interactive access Chameleon offers.
(2) The Recent-Deployed preemption policy is consistently the most energy-efficient as it yields the least HTC job reruns.
(3) Different algorithms exhibit different trade-offs: the most energy-efficient algorithm (Rolling-Median) has more sudden preemptions, while Rolling-Mean can provide more reliable advanced notifications to HTC with similar utilization improvement. The LSTM model overestimates the cloud user requests, and therefore, has more preemptions compared to the statistical models.

REFERENCES

[1] Jim Basney and Miron Livny. 1999. Deploying a High Throughput Computing Cluster. In *High Performance Cluster Computing: Architectures and Systems, Volume 1*, Rajkumar Buyya (Ed.). Prentice Hall PTR.

[2] Chameleon Cloud. [n.d.]. The Lease Calendars. https://chameleoncloud.readthedocs.io/en/latest/technical/reservations.html#the-lease-calendars.

[3] Science Clouds. [n.d.]. Cloud traces | Science Clouds - Cloud computing for science. https://www.scienceclouds.org/cloud-traces.

[4] Álvaro López García, Enol Fernández del Castillo, and Isabel Campos Plasencia. 2019. An efficient cloud scheduler design supporting preemptible instances. *Future Generation Computer Systems* 95 (2019), 68–78.

[5] Kate Keahey, Jason Anderson, Zhuo Zhen, Pierre Riteau, Paul Ruth, Dan Stanzione, Mert Cevik, Jacob Colleran, Haryadi S. Gunawi, Cody Hammock, Joe Mambretti, Alexander Barnes, François Halbach, Alex Rocha, and Joe Stubbs. 2020. Lessons Learned from the Chameleon Testbed. In *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20)*. USENIX Association.

[6] Feng Liu, Kate Keahey, Pierre Riteau, and Jon Weissman. 2018. Dynamically negotiating capacity between on-demand and batch clusters. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 493–503.

[7] Paul Marshall, Kate Keahey, and Tim Freeman. 2011. Improving utilization of infrastructure clouds. In *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE, 205–214.

[8] Ashish Kumar Mishra, Brajesh Kumar Umrao, and Dharmendra K Yadav. 2018. A survey on optimal utilization of preemptible VM instances in cloud computing. *The Journal of Supercomputing* 74, 11 (2018), 5980–6032.

[9] Ruth Pordes, Don Petravick, Bill Kramer, Doug Olson, Miron Livny, Alain Roy, Paul Avery, Kent Blackburn, Torre Wenaus, Frank Würthwein, Ian Foster, Rob Gardner, Mike Wilde, Alan Blatecky, John McGee, and Rob Quick. 2007. The open science grid. In *J. Phys. Conf. Ser. (78, Vol. 78)*. 012057. https://doi.org/10.1088/1742-6596/78/1/012057