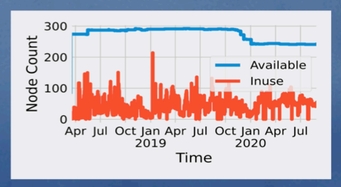
Yin and Yang: Balancing Cloud Computing and HTC Workloads

Zhuangwei Kang, Zhuo Zhen (advisor), Kate Keahey (advisor)

With the help of proper preemption policies and proactive resource schedulers, combining academic cloud and High Throughput Computing (HTC) systems through preemptible instances would help increase the utilization rate of the clouds and reduce the energy cost at the same time. We propose a data-driven simulator, then with which we evaluate a comprehensive set of resource scheduling strategies (4 preemption policies × 3 cloud user requests forecasting models).

Background

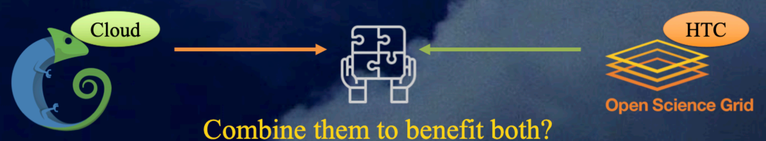
Cloud resources for CS systems research need to be available interactively to support exploration; this may result in **resources underutilization.**



Utilization of ComputeHaswell Node in Chameleon

HTC involves running many independent tasks implementing a domain science application that does not require interactivity and resilient to resource loss.

Problem with HTC: tasks that don't run to completion are simply re-executed, which may **waste time and energy**.



Combine them to benefit both?

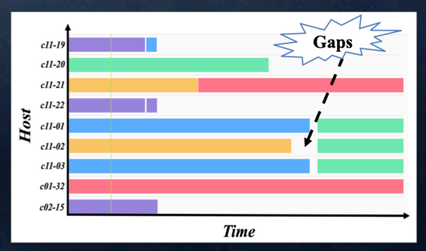
Analogy

Commercial cloud providers use preemptible instances to solve a similar problem.

AWS Spot VMs

GCP Preemptible VMs

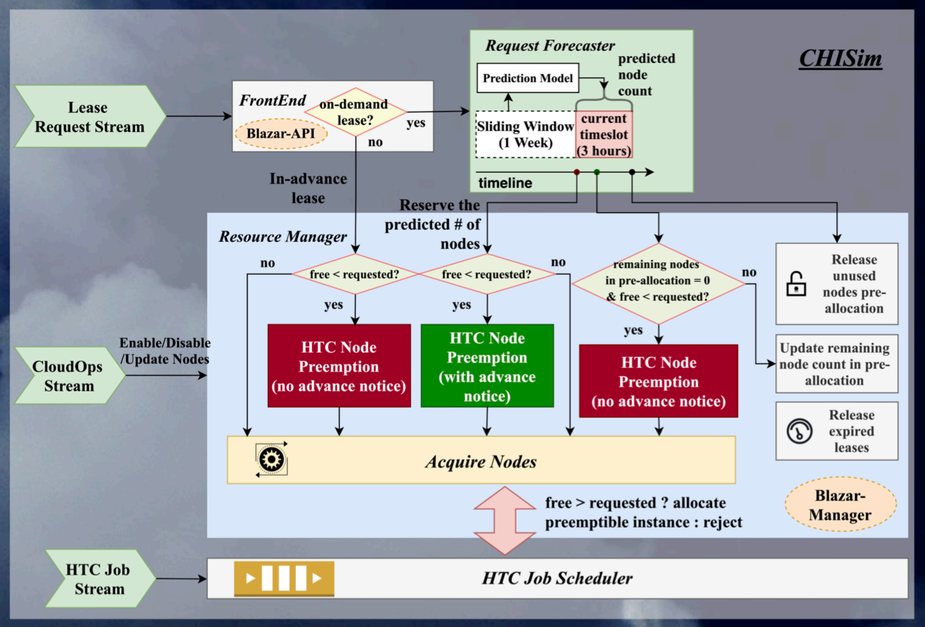
Problem Statement



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

- Can we fill these lease gaps by deploying HTC jobs?
- What is the most efficient way to run HTC jobs on preemptible instances?
- How can we minimize the cost of running HTC jobs by reducing the need to re-run them?

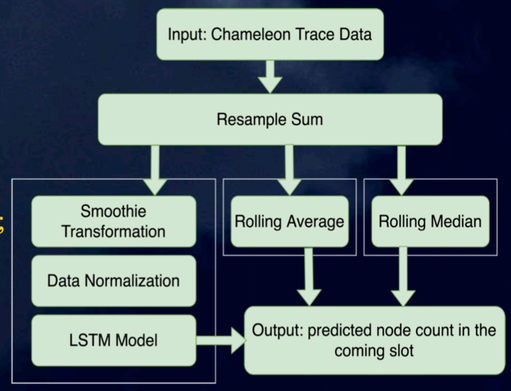
Approach



Preemption Policies

- **Random**: arbitrary HTC nodes;
- **Recent-Deployed**: most-recent deployed HTC nodes;
- **Least-resource(core) Used**: HTC nodes with the least number of cores;
- **Least-Resubmissions**: HTC nodes with the least number of resubmissions

Request Forecaster



Evaluation

Design of Experiments

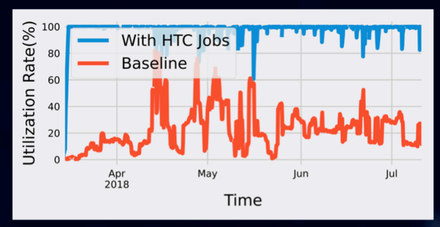
Simulation data: 3000 Chameleon leases (*ComputeHaswell Node*) and an HTC job log file.

Experiment	Description	Advance Notice
Baseline	Run Chameleon user requests only	No
Greedy Algorithm	Run HTC jobs on preemptible instances whenever they are available	No
Predictive Filling	Predict Chameleon user requests and manage the deployment of preemptible instances to reduce preemptions	Yes

Repeat Experiments

Learn More (In Progress)

Resource Utilization



Resource Utilization Rate (preemption policy: Recent-Deployed, Request Forecaster: Rolling-Median)

Predictive filling leads to trivial utilization degradation (0.21%-1.22%) compared to the greedy algorithm while increasing by 74.4%-75.9% relative to the Baseline (Chameleon-only). Rolling-Median introduces the lowest overhead (0.21%-0.45%).

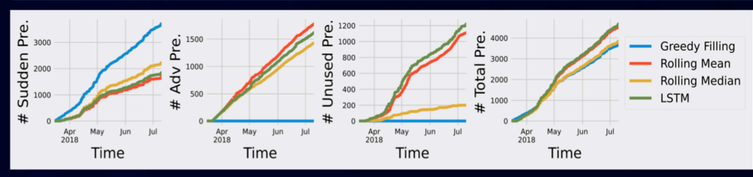
Energy Waste Rate

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

Mean Energy Waste Rate: The percentage of wasted core hours of completed jobs to the total core hours. (shadowed cell indicates the winner)

HTC Node Preemption

Preemption Types	Predict vs. Actual	Advance Notice	# of Preemptions
Sudden	<	No	Reduce 42.15% ~ 54.9% comparing to the Greedy filling algorithm
Unused	>	Yes	LSTM > Rolling-Mean > Rolling-Median > Greedy Filling
Advance	=	Yes	LSTM > Rolling-Mean > Rolling-Median > Greedy Filling



HTC Node Preemptions (policy: Recent-Deployed)

Conclusion

- Combining Chameleon and HTC workloads can increase utilization without compromising the interactive access Chameleon offers.
- The Recent-Deployed preemption policy is the most energy-efficient as it yields the least HTC job re-runs.
- 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.

Acknowledgements

NSF