VirginiaTech Invent the Future | HHMI janelia farm research campus | Los Alamos NATIONAL LABORATORY EST. 1943 | THE UNIVERSITY of NEW MEXICO

Poor Man's Social Network

Consistently Trade Freshness For Scalability

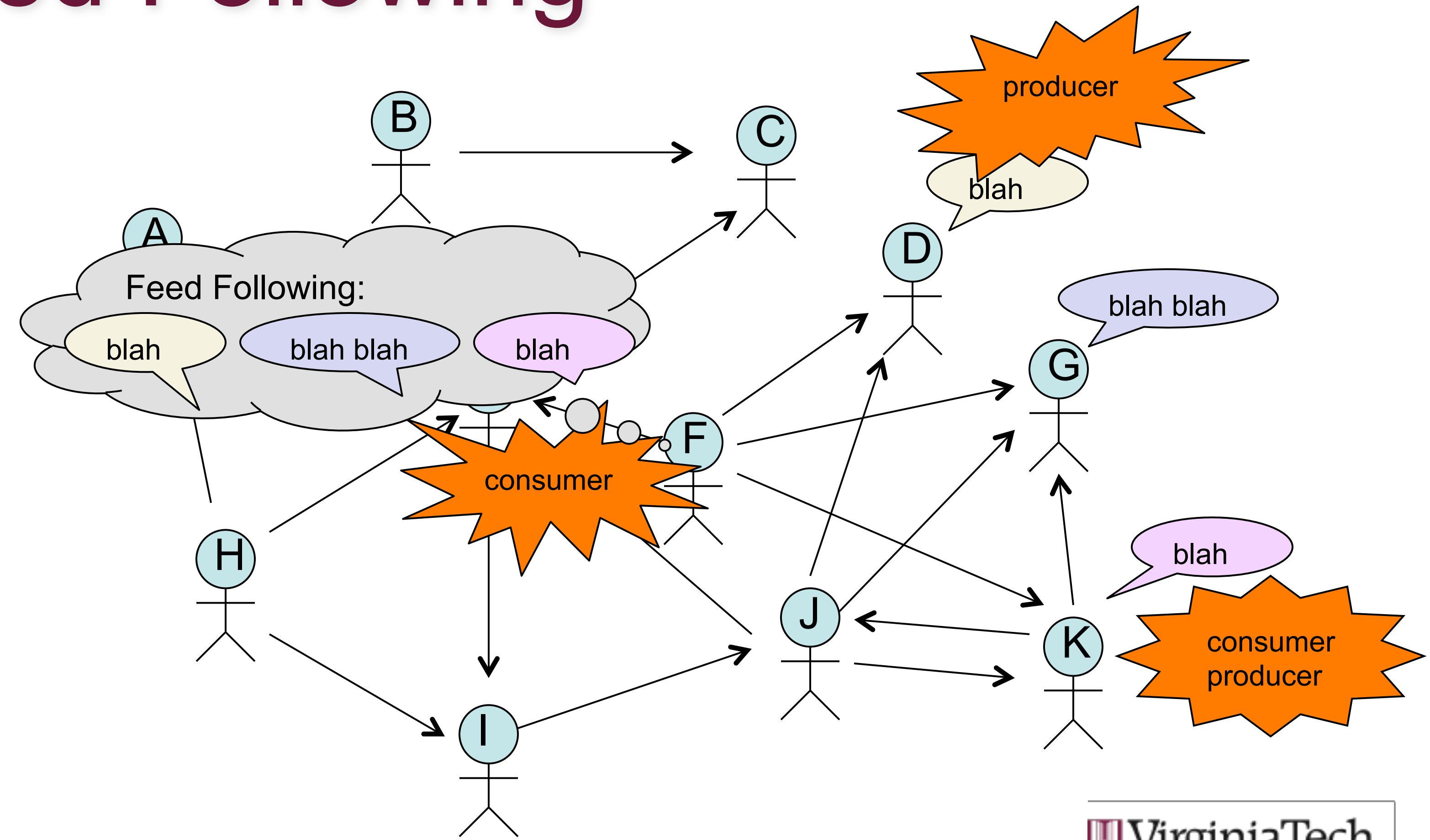
Zhiwu Xie, Jinyang Liu, Herbert Van de Sompel,
Johann van Reenen and Ramiro Jordan

Outline

- Scaling feed following
- Algorithm
- Experiment and results
- Conclusions

VirginiaTech
Invent the Future

Feed Following



VirginiaTech
Invent the Future

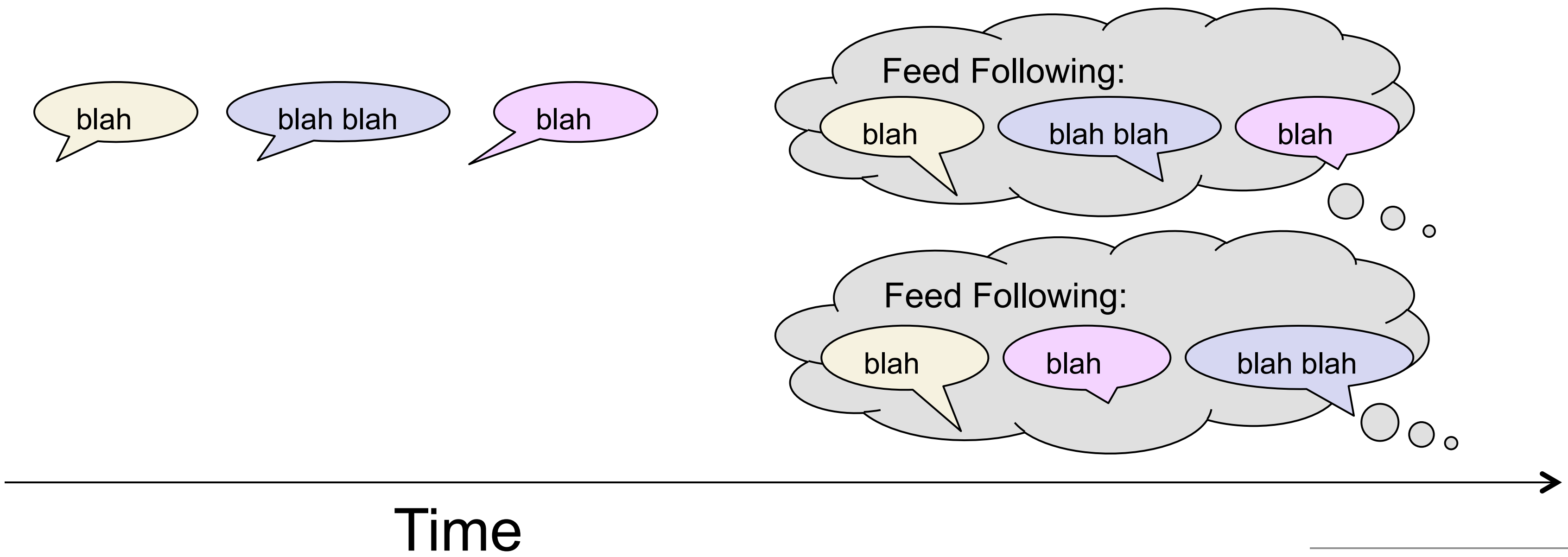
Feed Following Scalability

Give me the 20 ***most recent*** tweets sent by ***all*** the people ***I*** follow

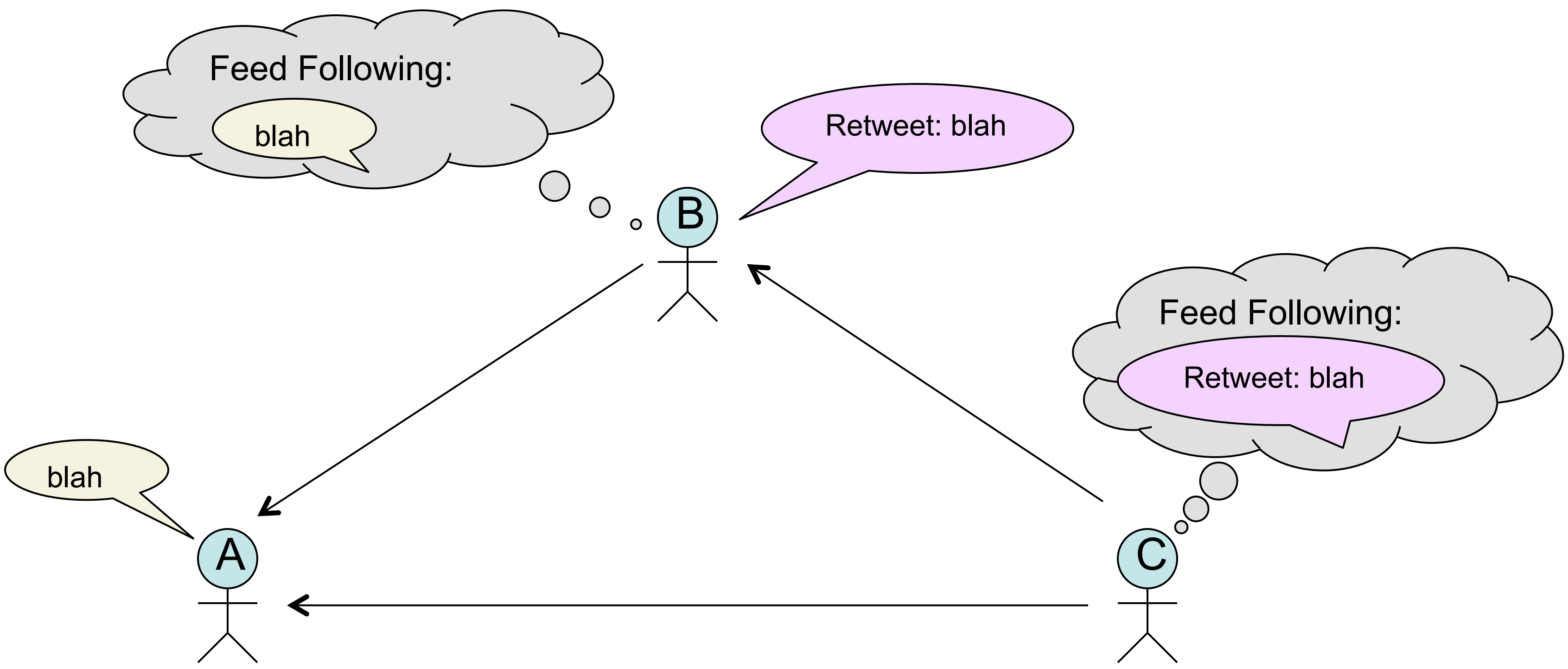
- Individualized queries
- Fast changing global state
- Partitioning, replication, and caching
- NoSQL: trade consistency for scalability

Consistency

- Atomicity, Linearizability, or One-copy Serializability (1SR)



Retweet Anomaly



New Approach: TimeMap Query

Who have created new tweets during the ***past scheduled release*** periods?

- Global time across partitions
- Schedule releasing
- Client-side processing and caching
- Consistently trade freshness for scalability

VirginiaTech
Invent the Future

CAP Theorem

- Preconditioned on the asynchronous network model: the only way to coordinate the distributed nodes is to pass messages
- In the partially synchronous model, where global time is assumed to be available, CAP may indeed be simultaneously achievable most of the time

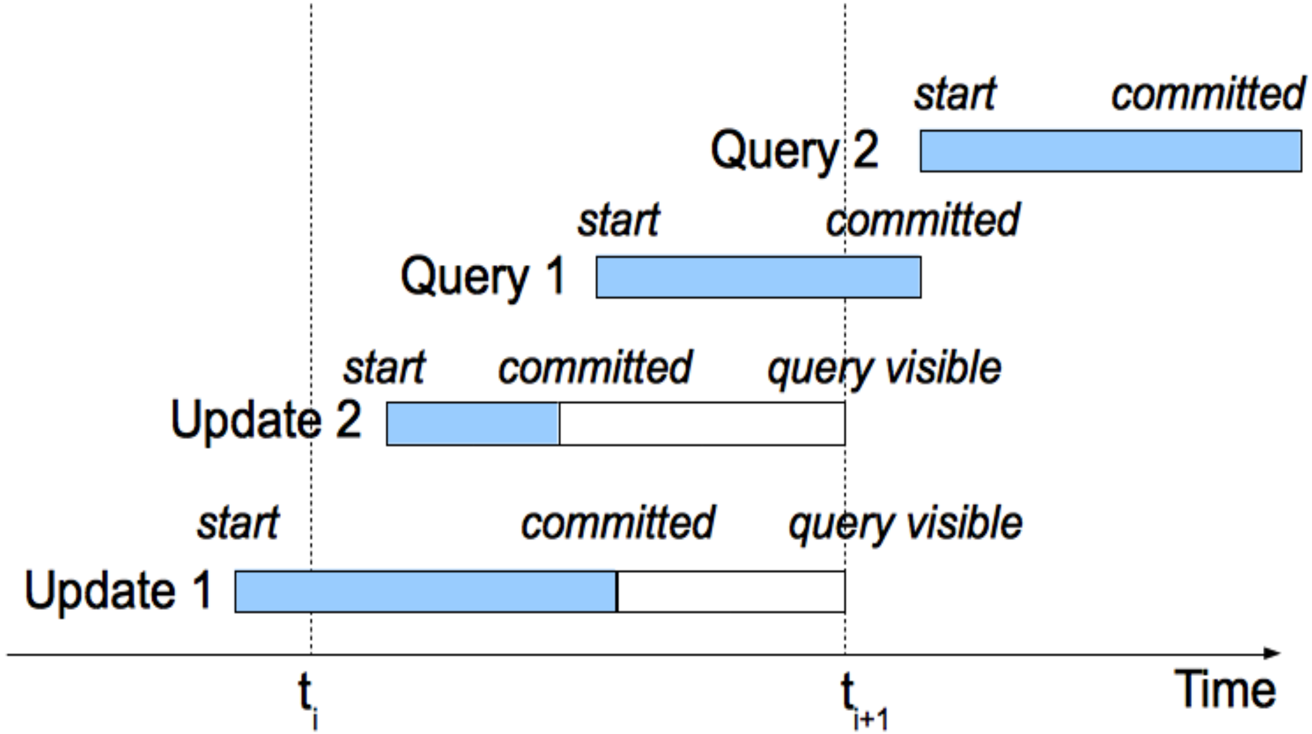
VirginiaTech
Invent the Future

Global Time

- "One of the mysteries of the universe is that it is possible to construct a system of physical clocks which, running quite independently of one another, will satisfy the Strong Clock Condition."

  – Time, Clocks and the Ordering of Events in a Distributed System, by Leslie Lamport

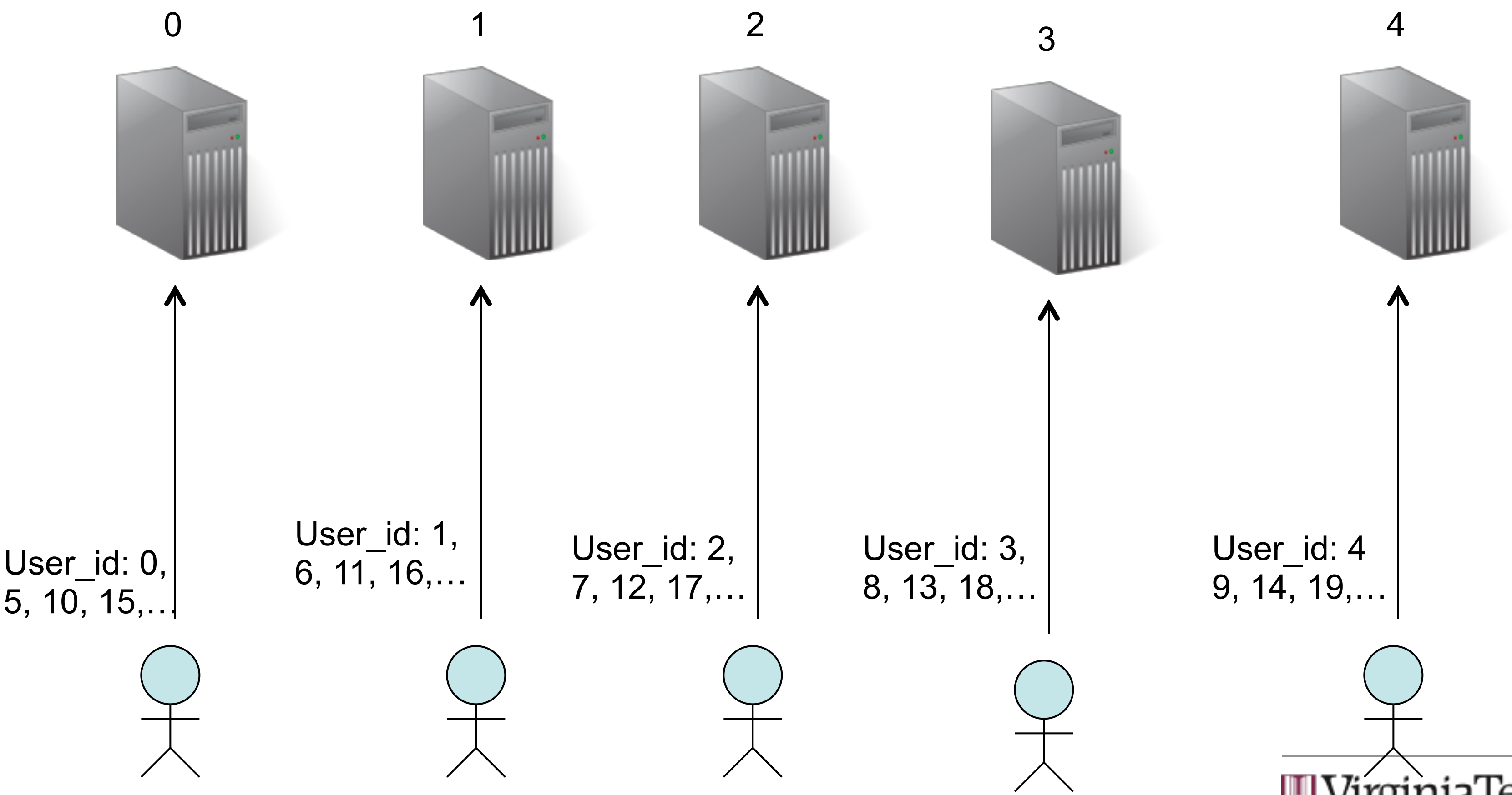
Scheduled Release Algorithm



Who have created new tweets during the ***past scheduled release*** periods?

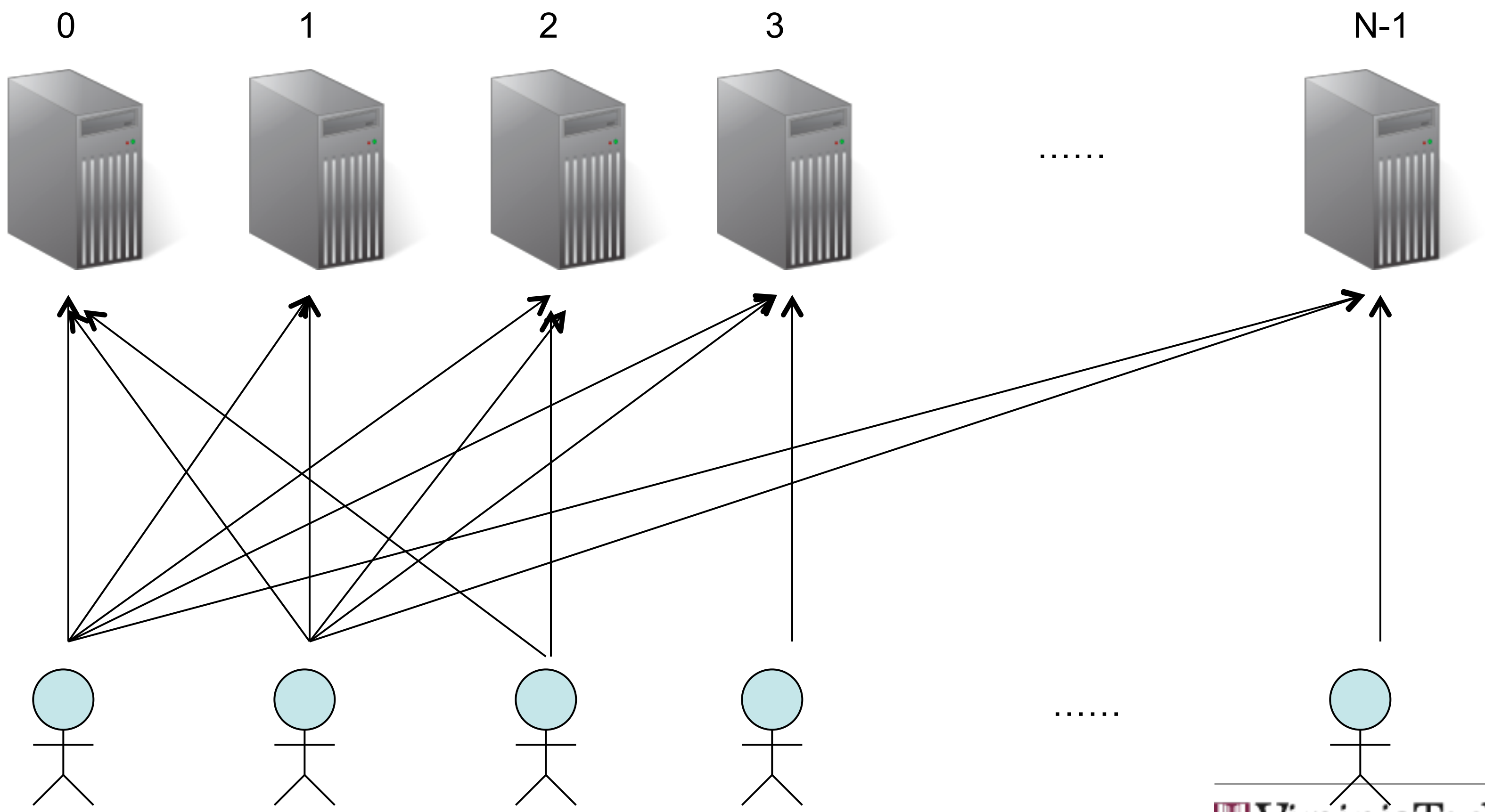
VirginiaTech Invent the Future

Partitioning: Send A New Tweet



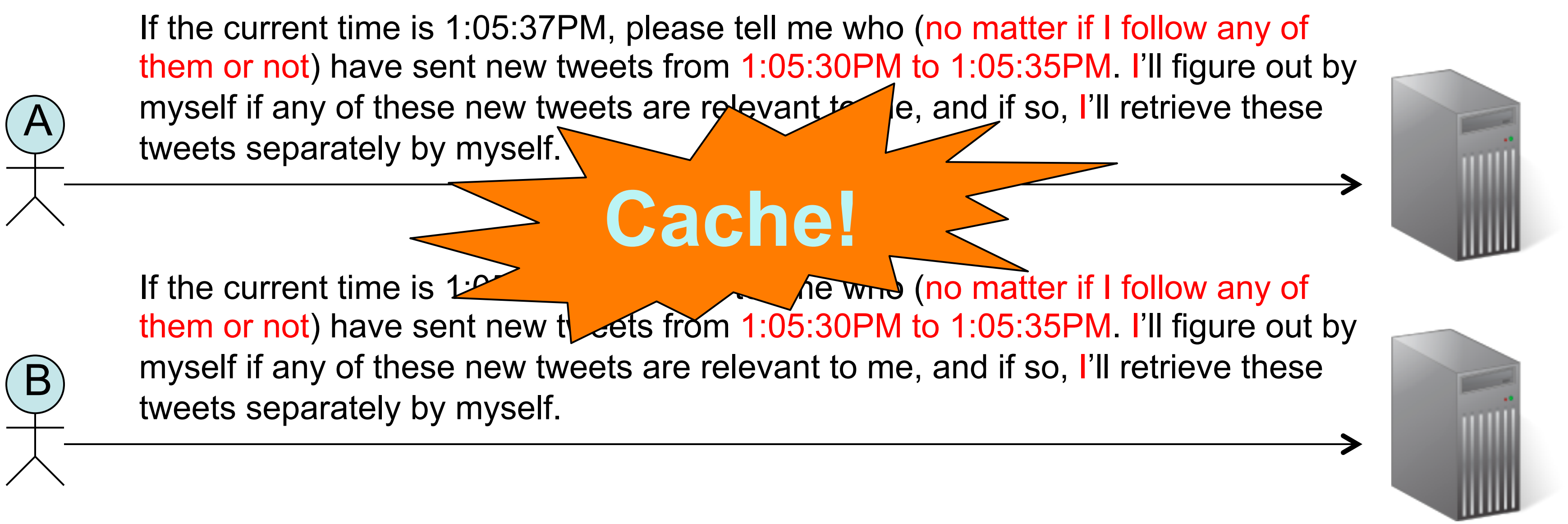
VirginiaTech
Invent the Future

Partitioning: TimeMap

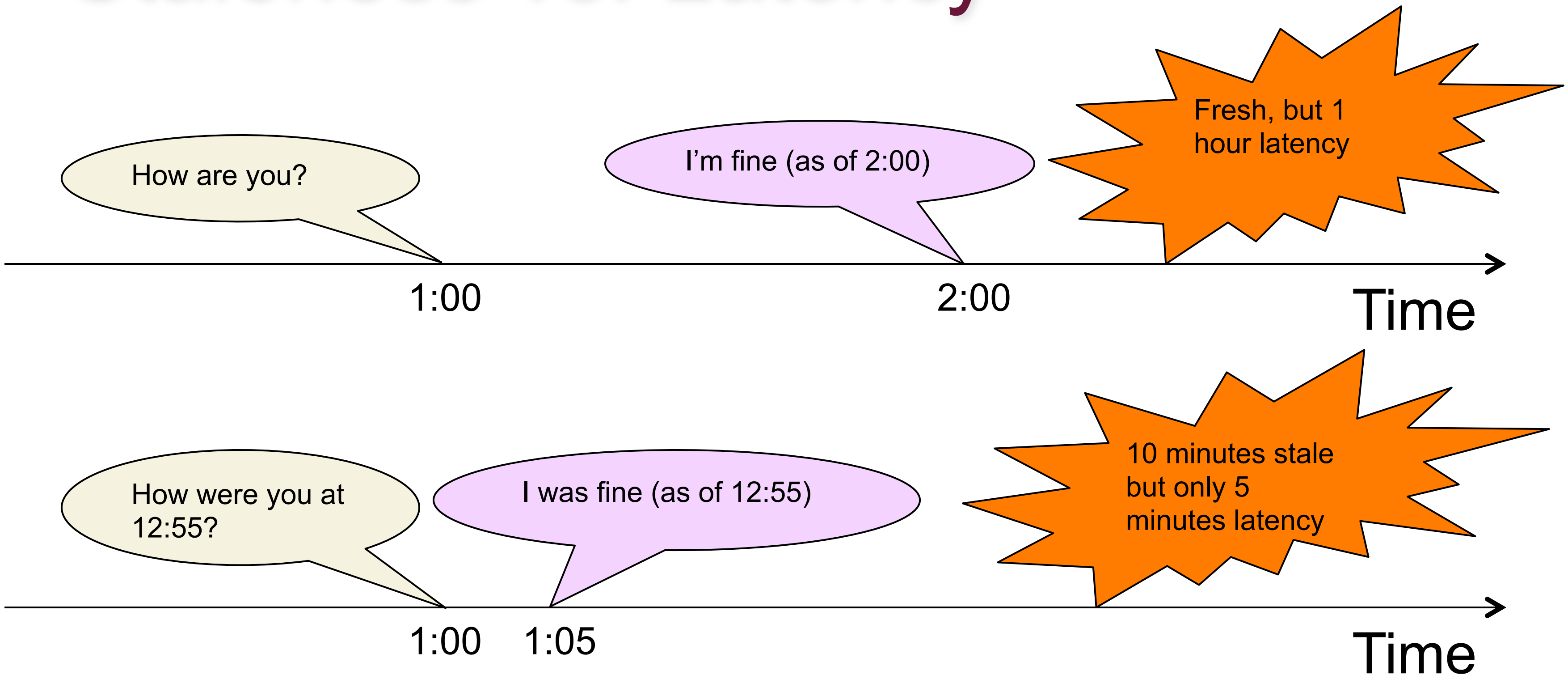


VirginiaTech
Invent the Future

Client Side Processing



Staleness vs. Latency



Trade Freshness For Scalability

- Mass transit system vs. private car
- Lose flexibility, but gain overall efficiency by sharing resources
- Stale up to the length of the schedule release period, e.g., 5 seconds.

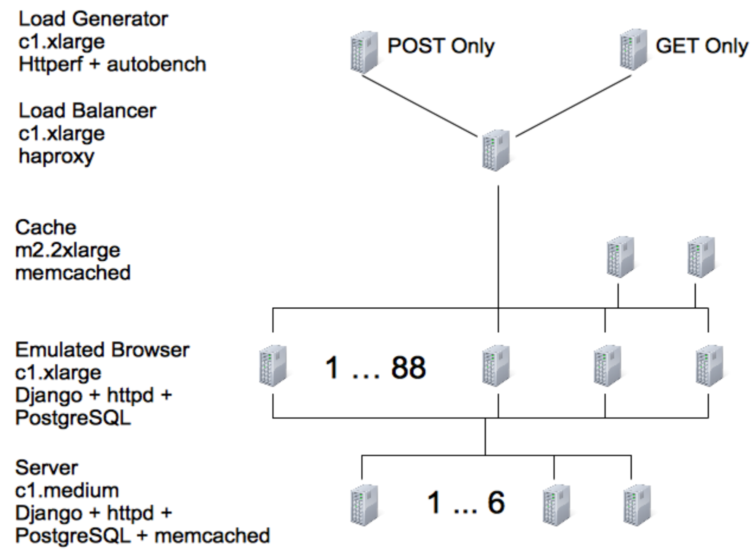
Experiment

- Implemented on AWS
- A Twitter like feed following application
- Server side: Python/Django, PostgreSQL, PL/pgSQL
- Client side: emulated browser, implemented in Python/Django and PostgreSQL

VirginiaTech
Invent the Future

Experiment: Configurations

- Used ~ 100 cloud instances from Amazon
- Most are used for emulated browsers
- 3 to 6 c1.medium as servers
- Use memcached to simulate caches



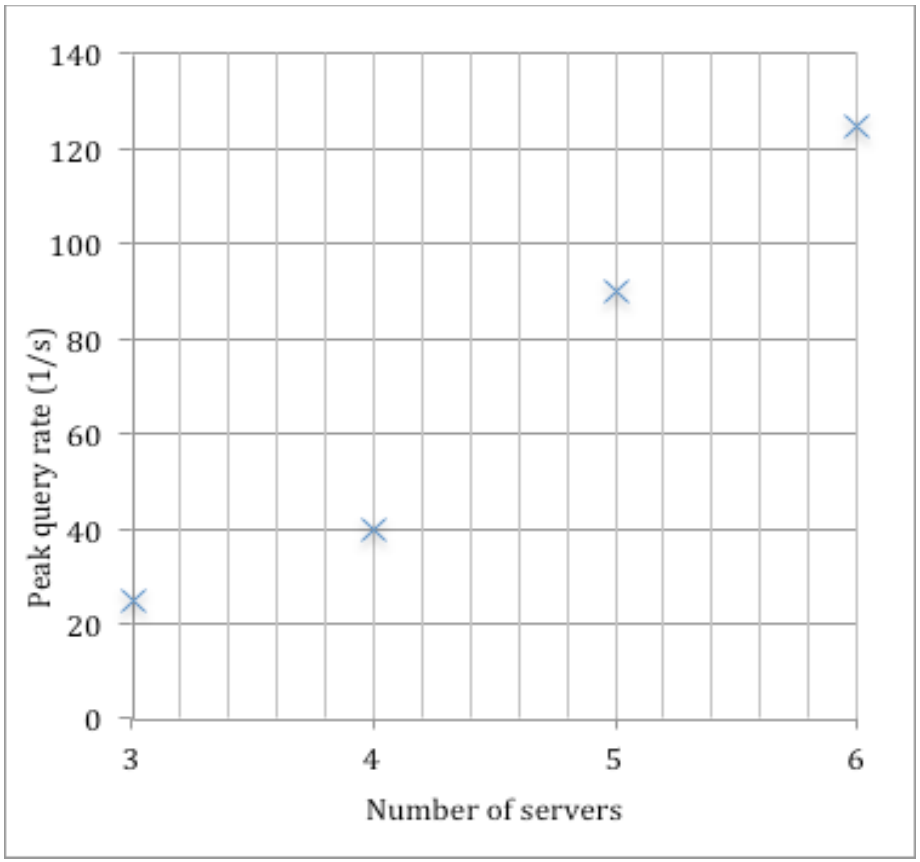
VirginiaTech
Invent the Future

Experiment: Workload

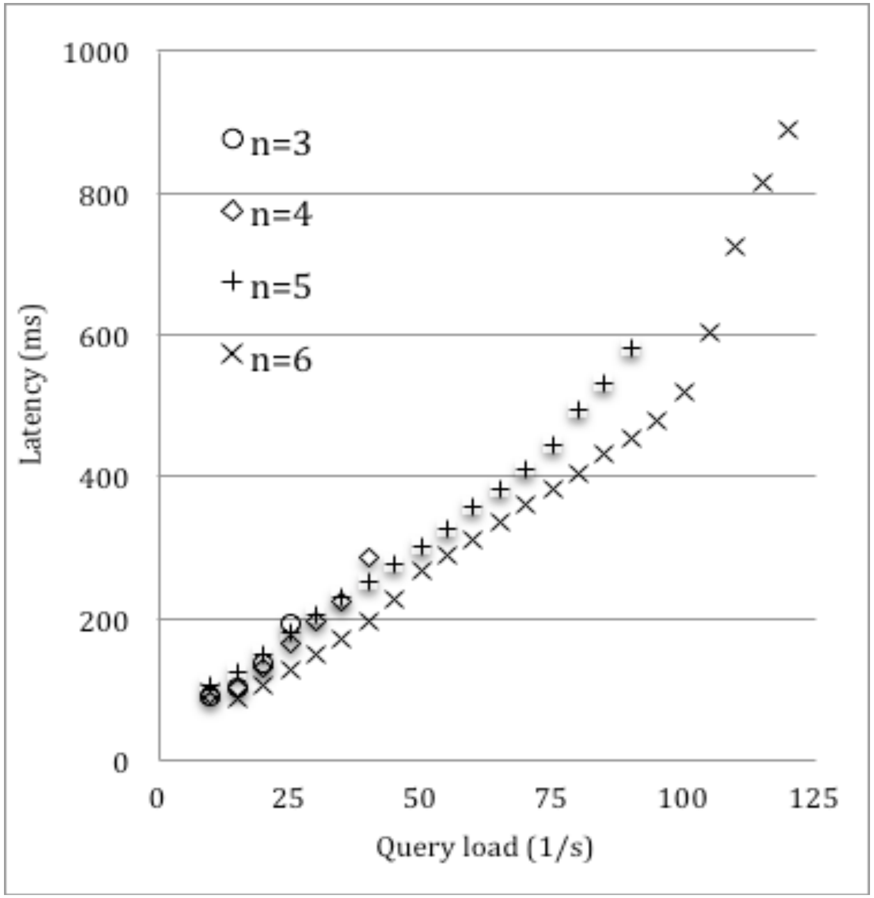
- Work load similar to the Yahoo! PNUTS experiment
- A following network of ~ 200,000 users
- Synthetic workload generated by Yahoo! Cloud Serving Benchmark

		PNUTS	This
Number of producers		67,921	67882
Number of consumers		200,000	196,283
Consumers per producer	Average	15.0	13.38
	Zipf parameter	0.39	0.39
Producers per consumer	Average	5.1	4.63
	Zipf parameter	0.62	0.62
Per-producer rate	Average	1/hour	1/hour
	Zipf parameter	0.57	0.57
Per-consumer rate	Average	5.8/hour	varied
	Zipf parameter	0.62	0.62

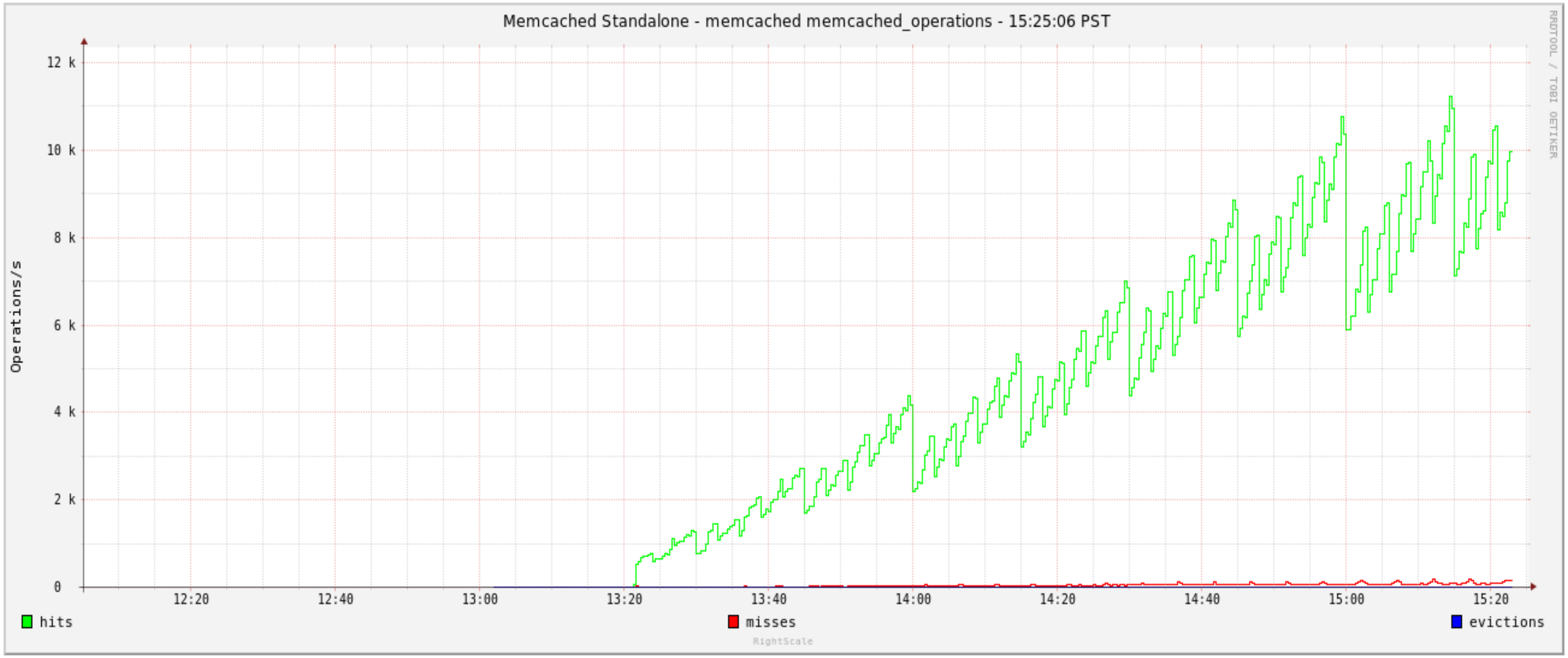
Experiment Result: Query Rate



Experiment Result: Latency

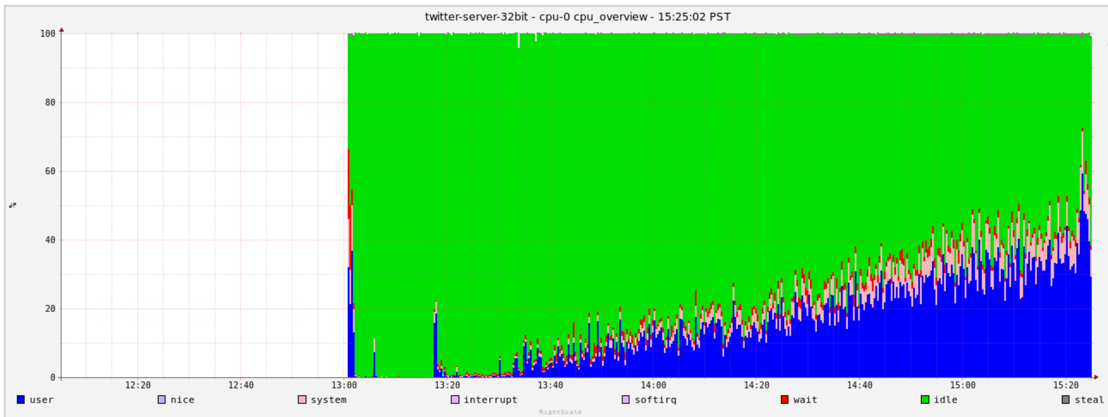


Experiment Results: Caching

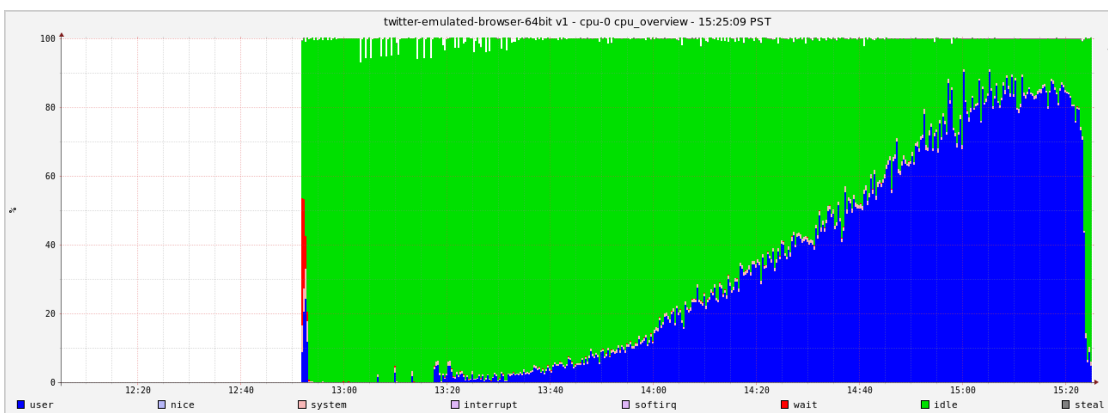


Experiment Results: CPU Load

Server



Client



VirginiaTech
Invent the Future

Conclusions

- Consistently scale feed following
- Linear scalability
- Practical low cost solution

Thank You

- Questions?