# Beaver:

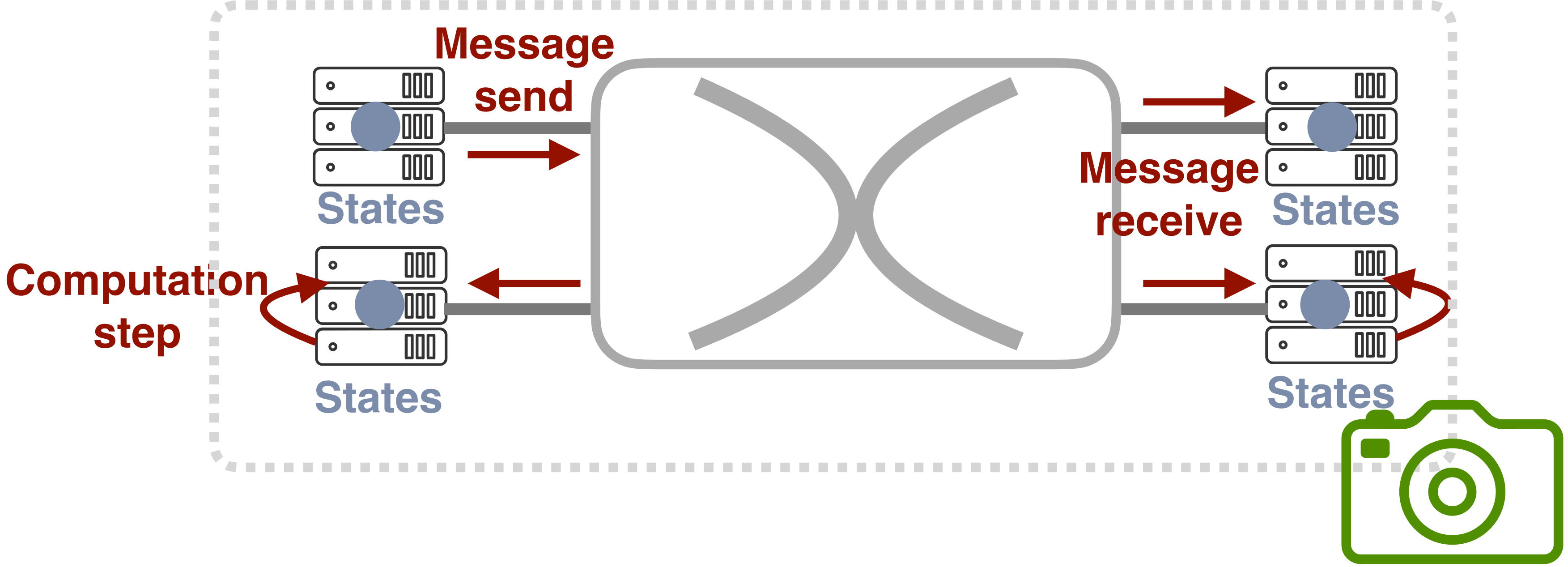
## Practical Partial Snapshots for Distributed Cloud Services

***Liangcheng (LC) Yu***, Xiao Zhang, Haoran Zhang, John Sonchack, Dan R. K. Ports, and Vincent Liu

University of Pennsylvania
SHANGHAI JIAO TONG UNIVERSITY
PRINCETON UNIVERSITY
Microsoft Research

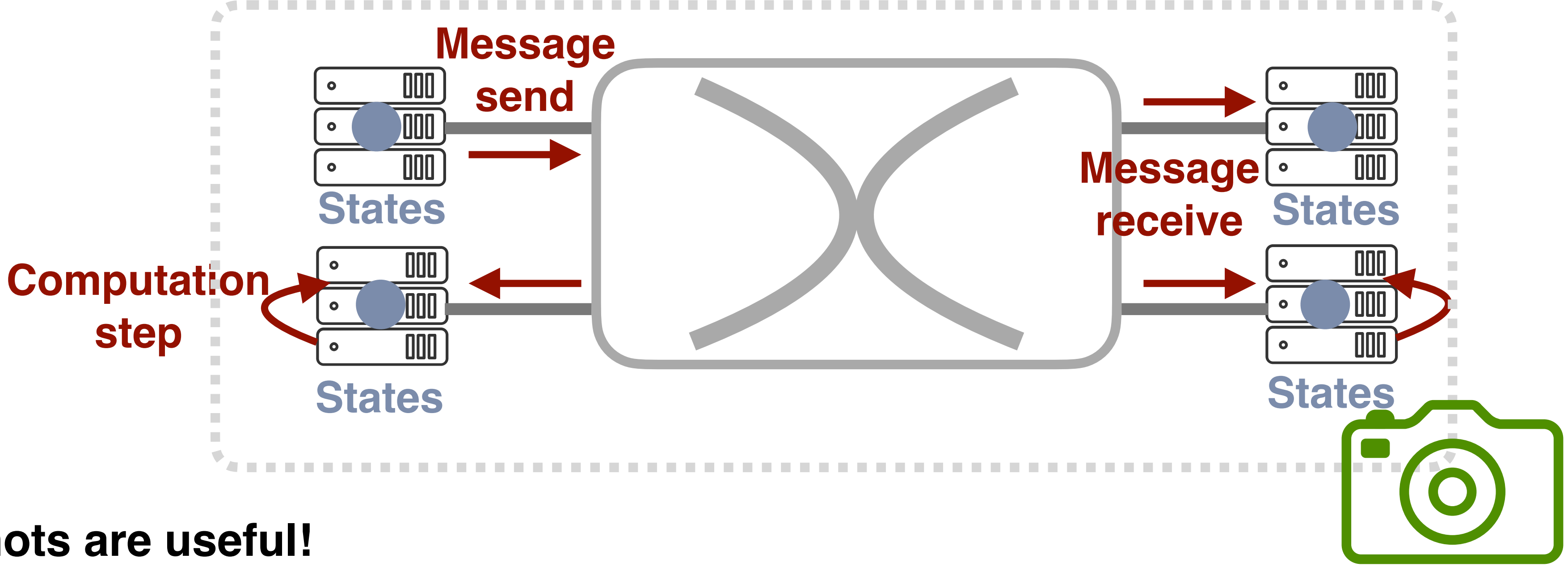
# Let's talk about snapshots

**Distributed snapshots:** a class of distributed algorithms to capture **consistent, global view** of **states**



# Let's talk about snapshots

**Distributed snapshots:** a class of distributed algorithms to capture **consistent, global view** of **states**




**Snapshots are useful!**

*Network telemetry*


*Distributed software debugging*

*Deadlock detection*


*Checkpointing and failure recovery*

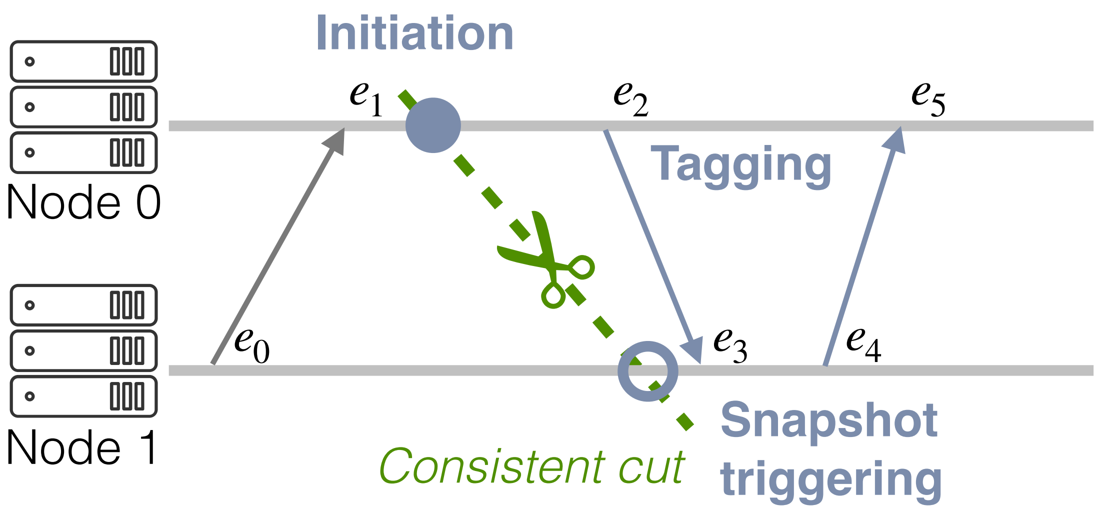
……

# Distributed snapshots 101

*A classic class of distributed protocols to capture a causally consistent view of states across machines.*

# Distributed snapshots 101

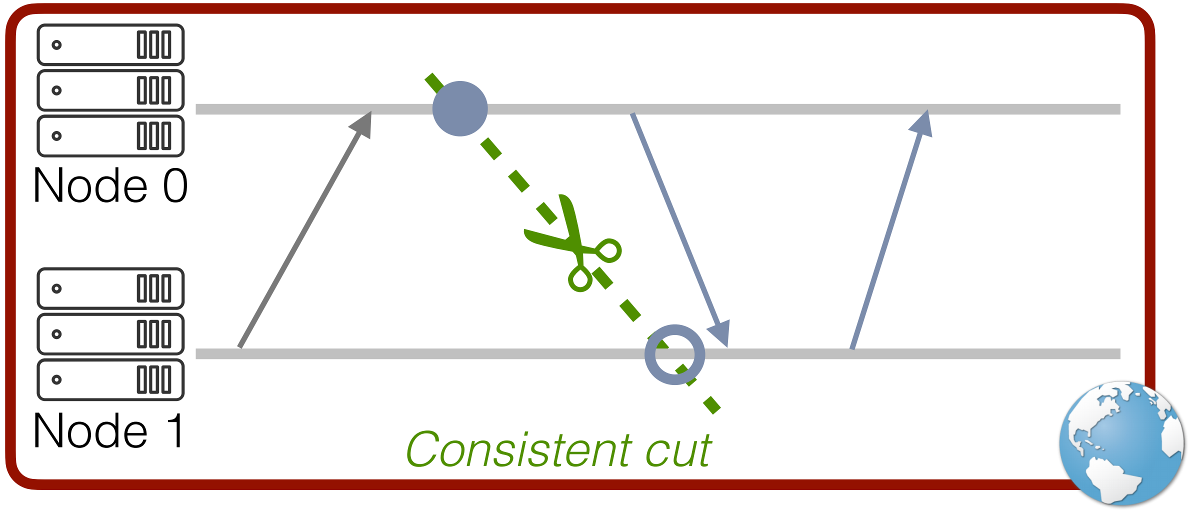
*A classic class of distributed protocols to capture a causally consistent view of states across machines.*




**Guarantee of causal consistency** ✅

*For **any** event $e$ in the cut, if $e' \rightarrow e$ (Lamport's 'happened before'), $e'$ is in the cut.*

# Are we done yet?

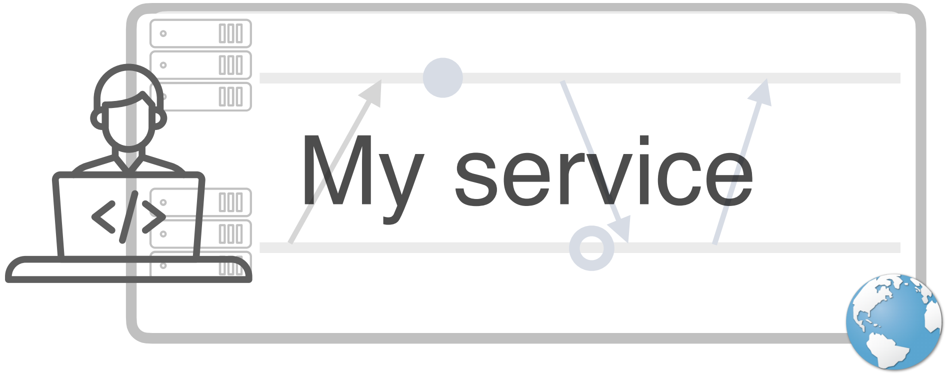



***Utopian: isolated 'universe' of nodes***

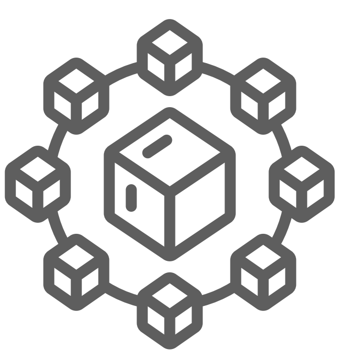
**Fundamental assumption**:
The set of participants are ***closed*** under causal propagation.

*Unfortunately, the assumption mismatches the real-world scenarios!*

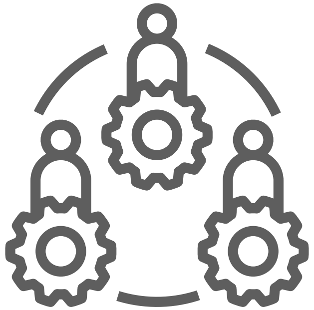
# The assumption mismatches the reality!




***Utopian: isolated ‘universe’ of nodes***

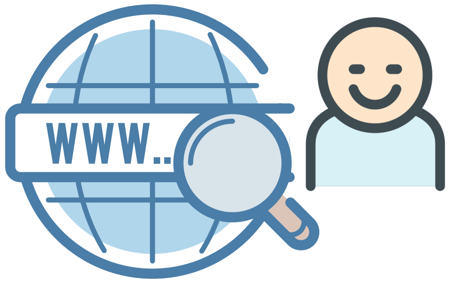


**Modular services**



**Instrumentation constraints**

**Costs and overheads**




**Hidden causality due to human**

# The assumption mismatches the reality!

Utopian: isolated ‘universe’ of nodes

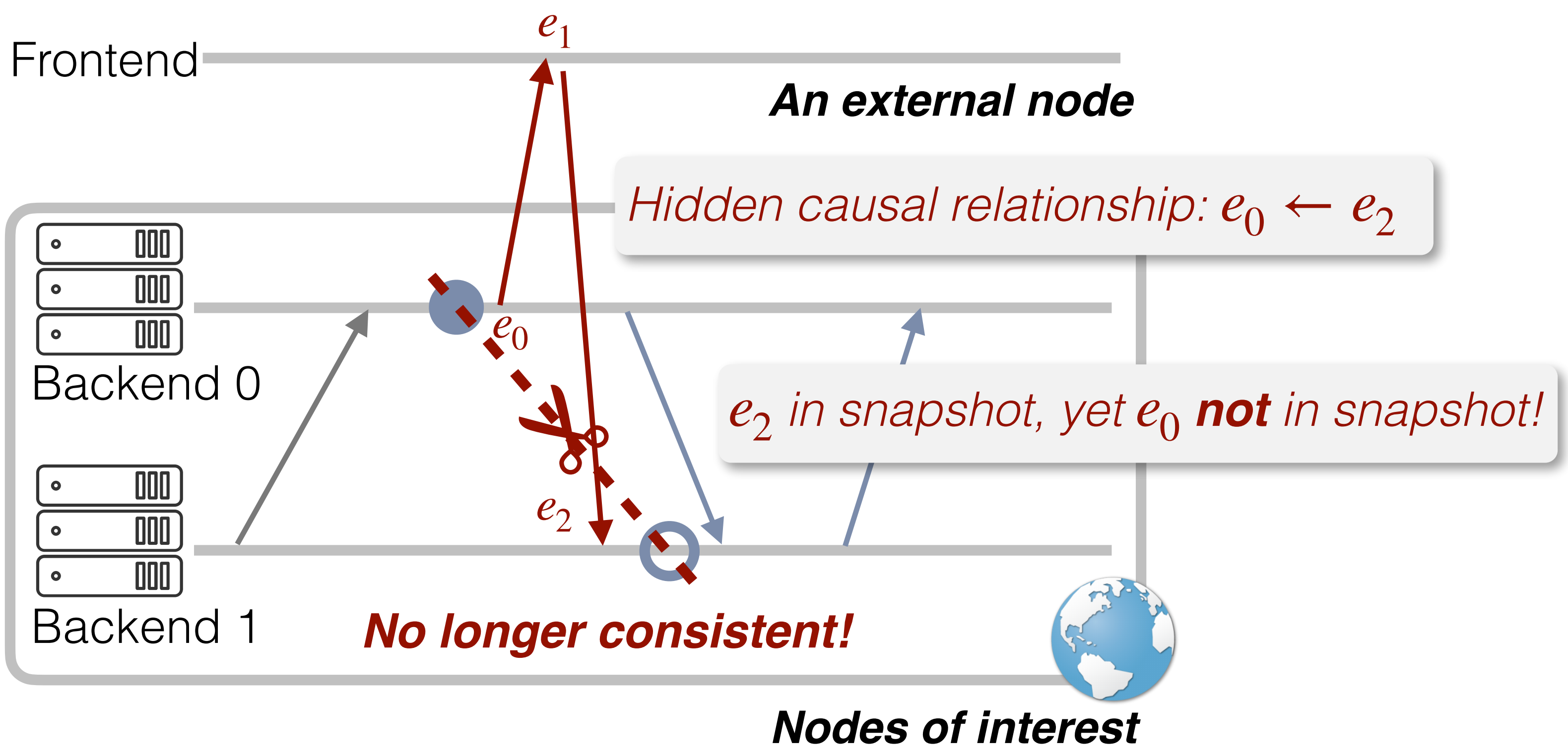
**Unrealistic** to assume *zero* external interaction

**Impractical** to instrument *all* processes


Costs and overheads

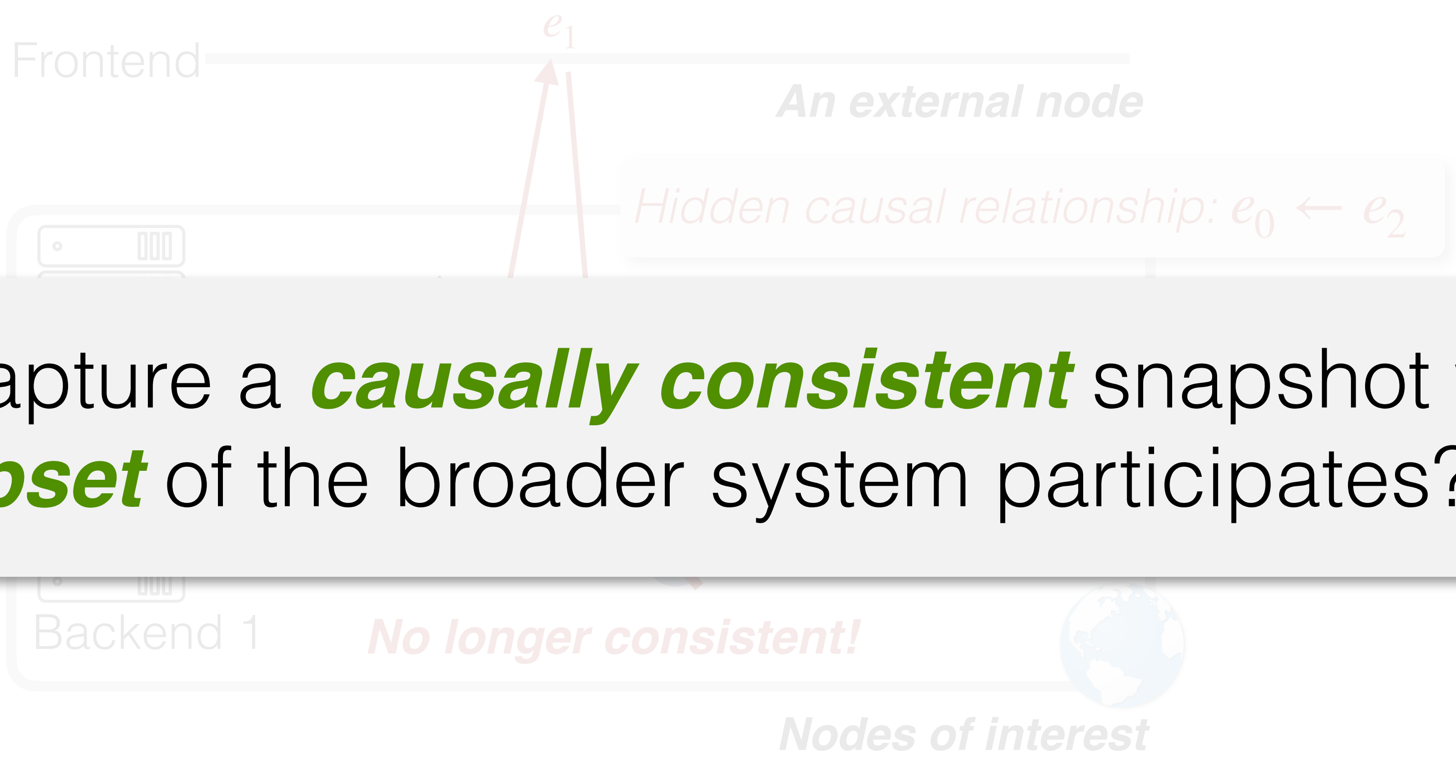
Hidden causality due to human

# Consequences?




☹ *A single external node can break the guarantee!*

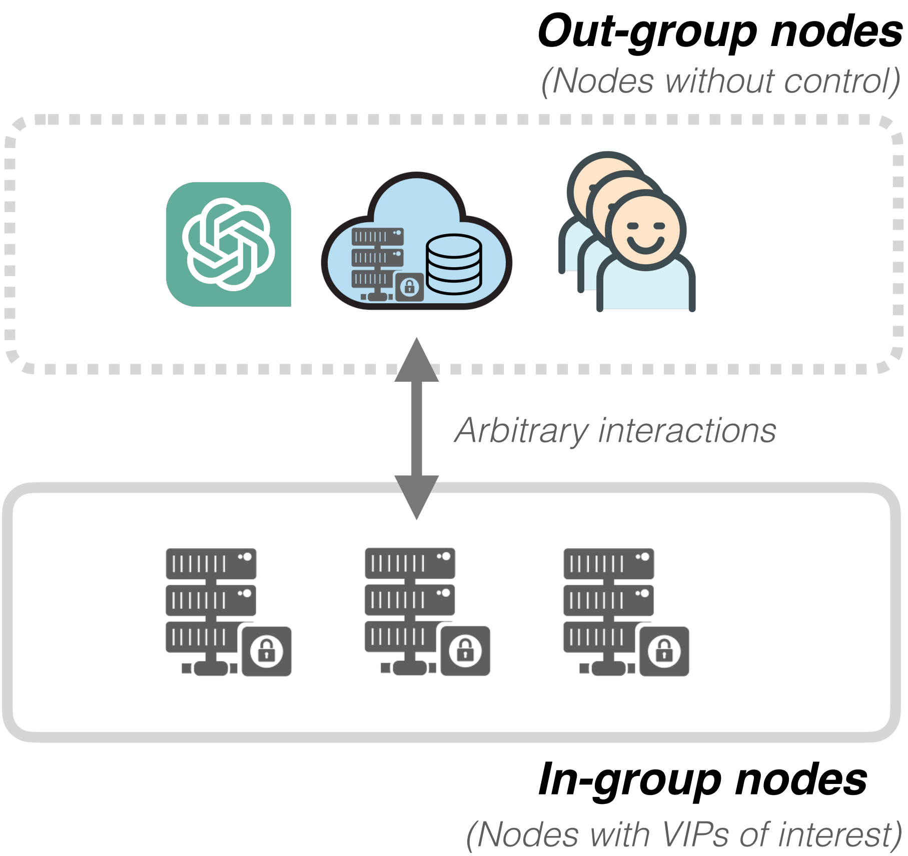
# Consequences?




Can we capture a ***causally consistent*** snapshot when a ***subset*** of the broader system participates?

*A single external node can break the guarantee!*

# Beaver: practical partial snapshots 




## The same causal consistency abstraction

*Even when the target service interact with **external, black box services** (arbitrary number, scale, placement, or semantics) via **arbitrary pattern** (including multi-hop propagation of causal dependencies)*

## Zero impact over existing service traffic

*That is, **absence of blocking or any form of delaying operations** during distributed coordination*

# Beaver: practical partial snapshots

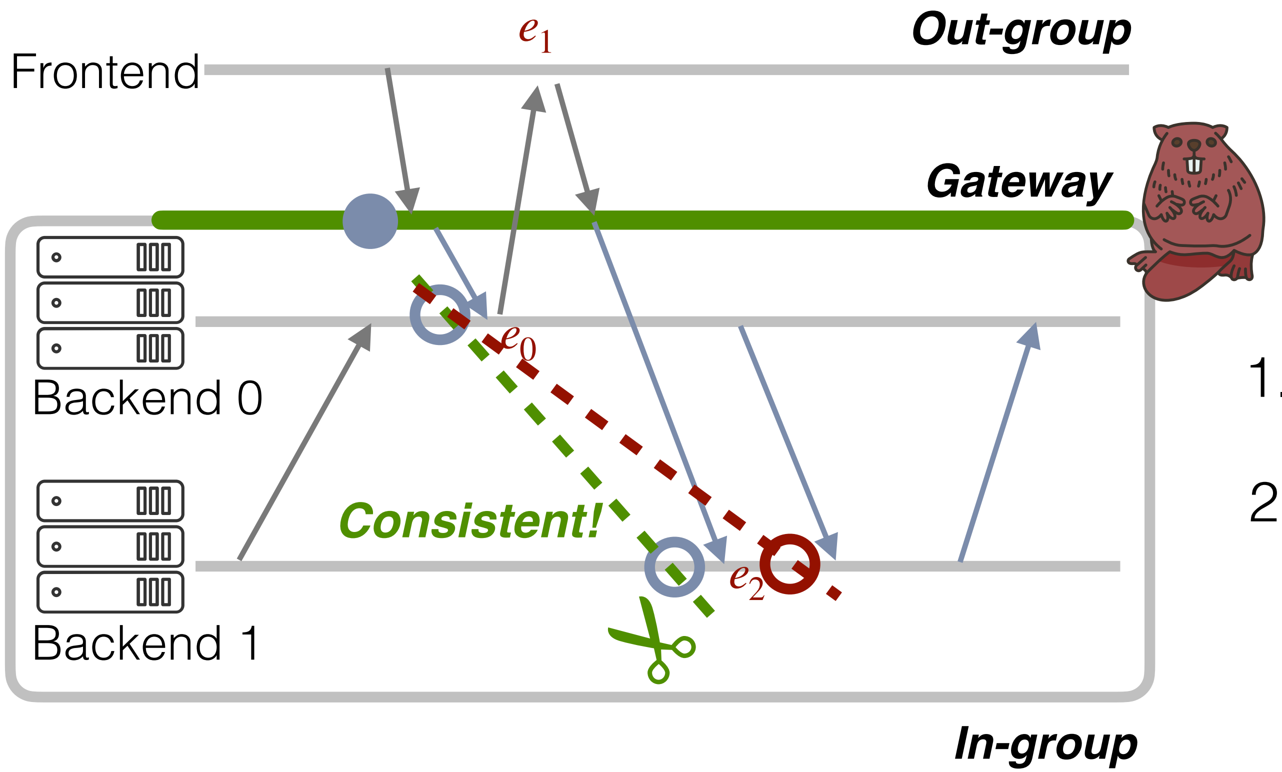
Out-group nodes
(Nodes without control)

How is it even possible ***without*** coordinating machines external to those of interest?

**Build a dam like a Beaver!**

In-group nodes
(Nodes with VIPs of interest)

# Idea 1: Gateway (GW) indirection




Beaver's gateway (GW) indirection:

1. Initiate GW to enter snapshot out-of-band
2. Mark ***inbound*** packets correspondingly

Before: **inconsistent** cut at O (after $e_2$)

With GW: **consistent** cut at O (before $e_2$)

# Formalizing idea 1: Monolithic Gateway Marking

***Theorem* 1.** *With MGM, a partial snapshot $C_{part}$ for $P^{in} \subseteq P$ is causally consistent, that is, $\forall e \in C_{part}$, if $e'.p \in P^{in} \wedge e' \rightarrow e$, then $e' \in C_{part}$.*

*Proof.* Let $e.p = p_i^{in}$ and $e'.p = p_j^{in}$. There are 3 cases:

1. Both events occur in the same process, i.e., $i = j$.
2. $i \neq j$ and the causality relationship $e' \rightarrow e$ is imposed purely by in-group messages.
3. Otherwise, the causality relationship $e' \rightarrow e$ involves *at least* one $p \in P^{out}$.

In cases (1) and (2), the theorem is trivially true using identical logic to proofs of traditional distributed snapshot protocols. We prove (3) by contradiction.

Assume $(e \in C_{part}) \wedge (\exists e' \rightarrow e)$ but $(e' \notin C_{part})$. With (3), $e' \rightarrow e$ means that there must exist some $e^{out}$ (at an out-group process) satisfying $e' \rightarrow e^{out} \rightarrow e$. Now, because $e' \notin C_{part}$, we know $e^{ss}_{p_j^{in}} \rightarrow e'$ or $e^{ss}_{p_j^{in}} = e'$, that is, $p_j^{in}$'s local snapshot happened before or during $e'$. Combined with the fact that the gateway is the original initiator of the snapshot protocol, we know that $e^{ss}_g \rightarrow e' \rightarrow e^{out} \rightarrow e$.

We can focus on a subset of the above causality chain: $e^{ss}_g \rightarrow e$. From the properties of the in-group snapshot protocol, $e^{ss}_g \rightarrow e$ implies that $e \notin C_{part}$.

This contradicts our original assumption that $e \in C_{part}$! $\square$

***Formal proof in paper***

Holds even if treating the out-group nodes as black boxes

Sufficient to ***only*** observe the inbound messages

# Key ideas in Beaver 

*How to ensure consistency without coordinating external machines?*

**Idea 1: Indirection through Monolithic Gateway Marking**

*How to instantiate the theoretical model in practice?*

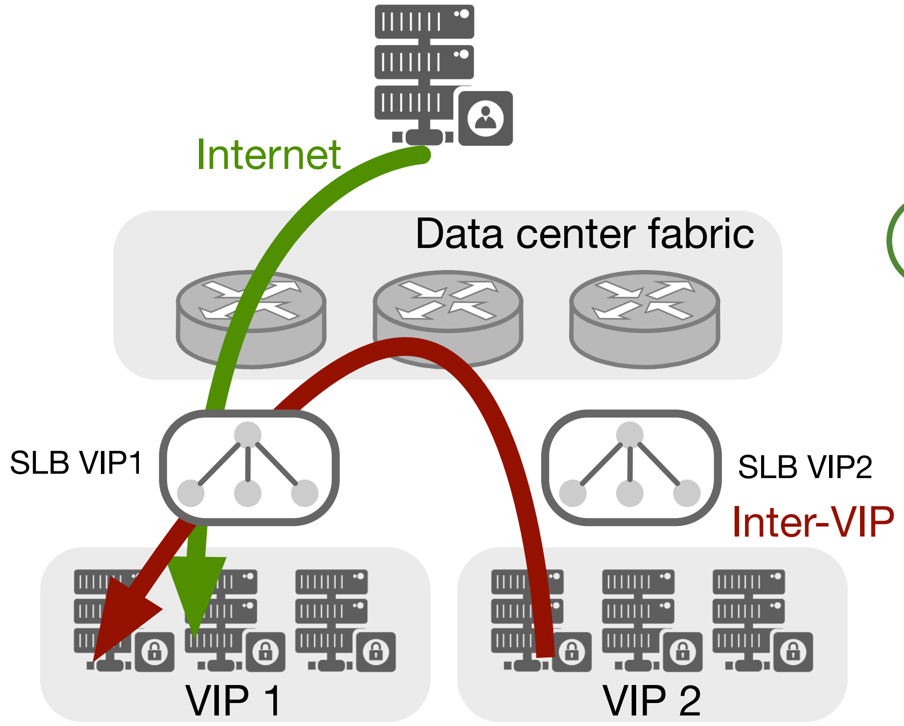
**Challenge 1** How to practically instantiate GW?

**Challenge 2** How to handle asynchronous GWs?

# Challenge 1 : instantiating GWs

☹ Rerouting all inbound traffic through the GW is ***costly***

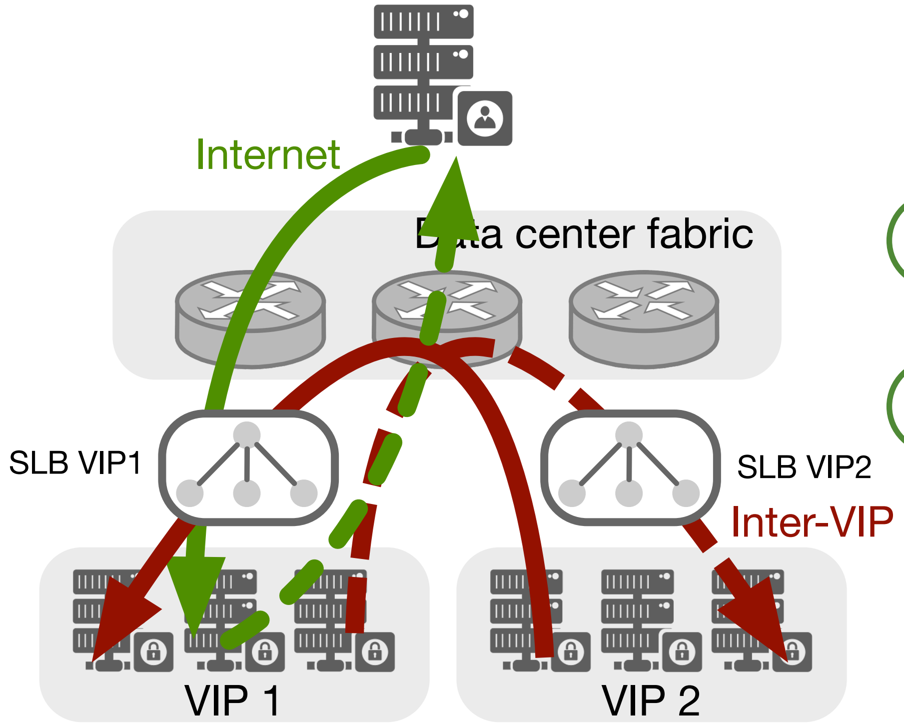
💡 Cloud data centers already place layer-4 load balancers (SLBs)




☺ SLBs as a natural candidate for in-situ marking

# Challenge 1 : instantiating GWs

- Rerouting all inbound traffic through the GW is ***costly***
- Cloud data centers already place layer-4 load balancers (SLBs)




- SLBs as a natural candidate for in-situ marking
- Beaver is compatible with SLB’s partial visibility due to DSR (Direct Server Return)

# Key ideas in Beaver 

*How to ensure consistency without coordinating external machines?*

**Idea 1: Indirection through Monolithic Gateway Marking**

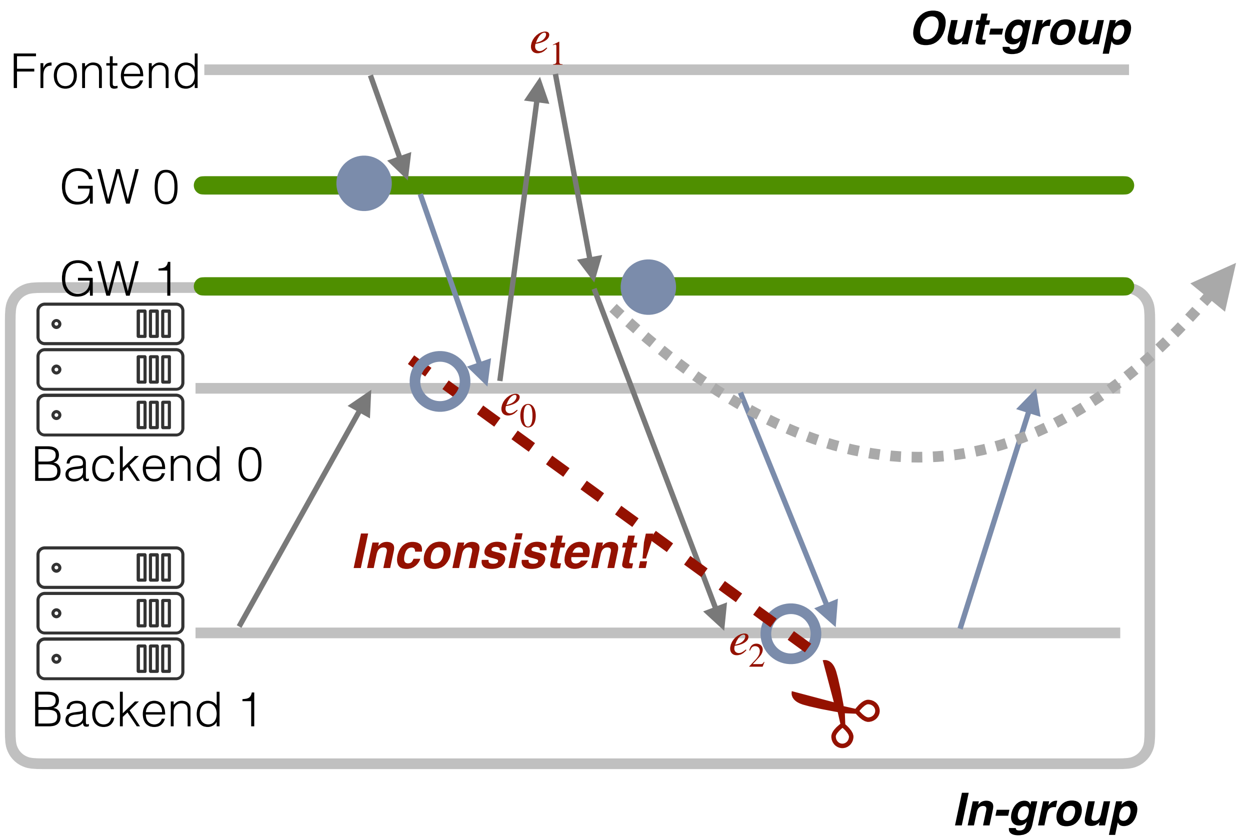
*How to instantiate the theoretical model in practice?*

**Challenge 1** How to practically instantiate GW?

**Idea 2: Exploit the unique location of existing SLBs**

**Challenge 2** How to handle asynchronous GWs?

# Implications of multiple SLBs

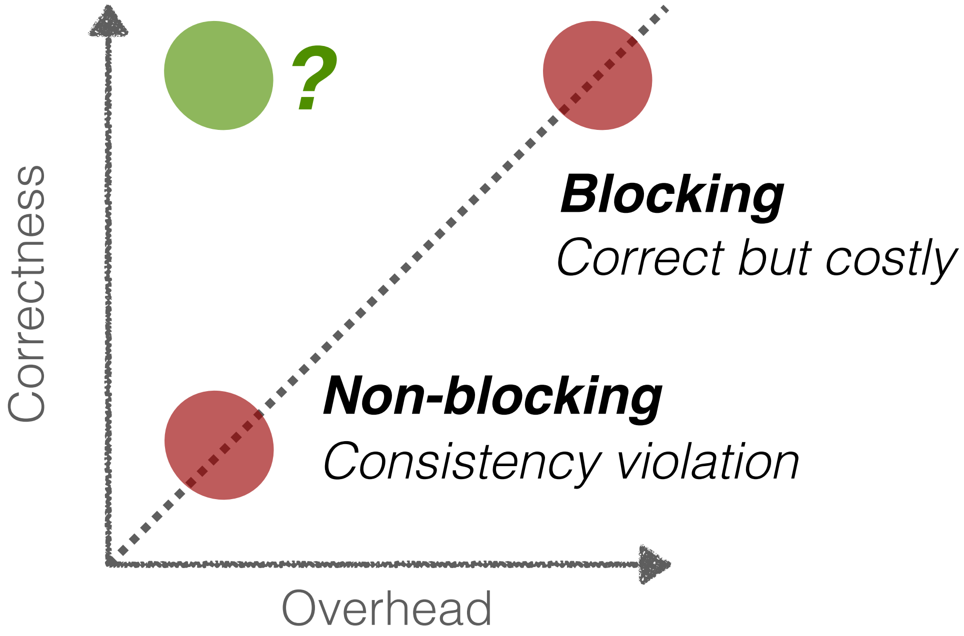



GW 1 hasn't initiated the new snapshot mode to mark it, triggering the ***violation***

***$e_2$ in snapshot, yet $e_0$ that leads to it is not, inconsistent!***

# Handling multiple GWs: design space

*How about blocking messages to 'atomically' trigger all SLBs?*



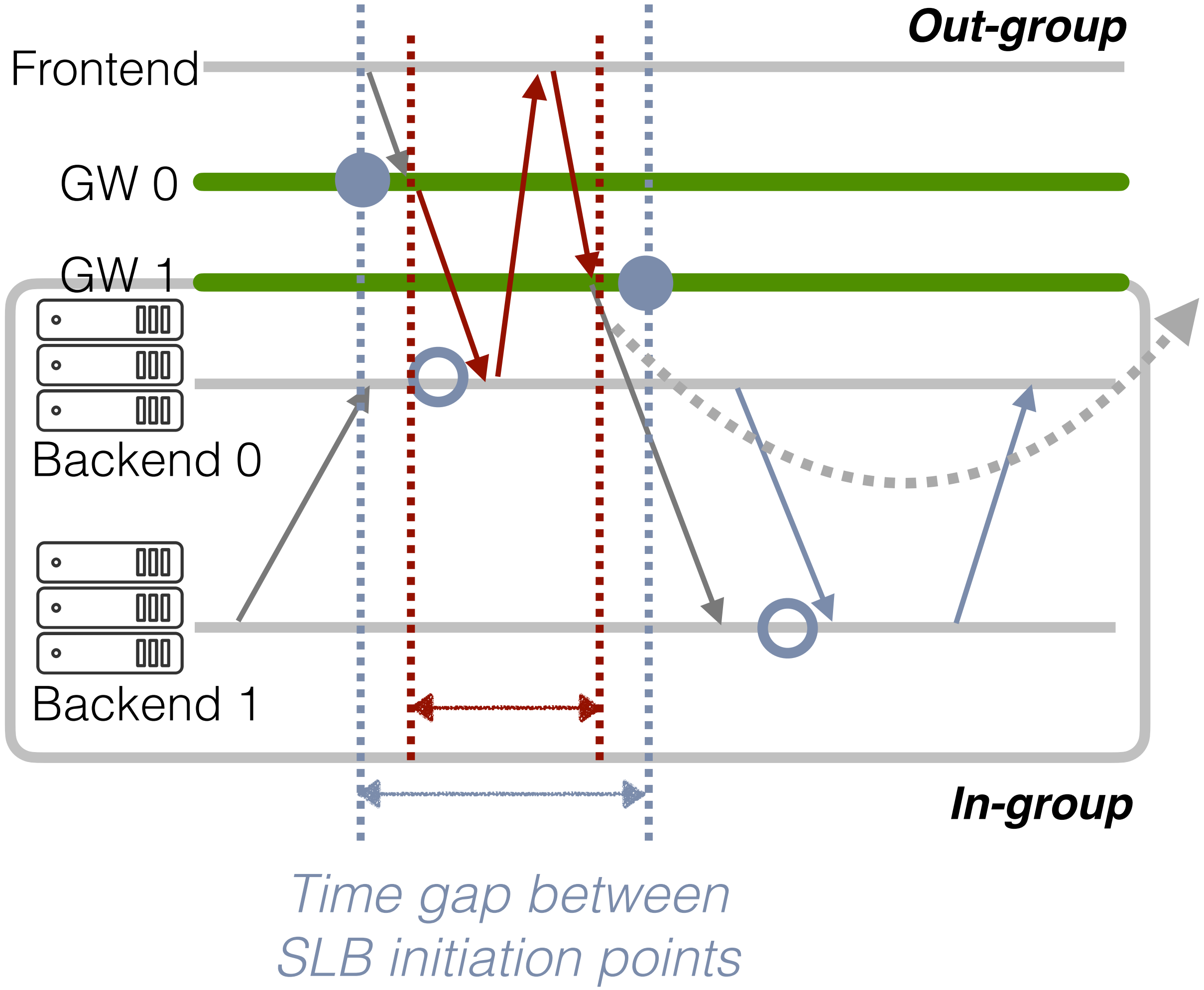

*Can we get both **consistency** and **zero impact** to service traffic (i.e., non-blocking)?*

**Optimistic Gateway Marking (OGM)**

- *Intuition & formalism*
- *Mechanism*

# Challenge 2: handling multiple SLBs

***Reflection***: *Beyond worst cases, when and how often does the violation occur?*




***Observation:***

***Causally relevant messages*** are rare!
GW→in-group→out-group→GW (external causal chain)

***Intuition***: *the resulting snapshot is consistent*

*1. if ↔ is **large enough***
*2. or if ↔ is **'close' enough***

**Theorem**: if ↔ < ↔, the partial snapshot is consistent!

↔ ≡ *Time gap between initiator-to-SLB one-way delays*

↔ ≡ *Time to form an external causal chain (GW→in-group→out-group→GW)*

***Theorem 2.*** *In a system with multiple asynchronous gateways, let the wall-clock time of the first and last gateway snapshots be $e^{ss}_{gmin} = \min_{e^{ss}_g}(e^{ss}_g.t)$ and $e^{ss}_{gmax} = \max_{e^{ss}_g}(e^{ss}_g.t)$, respectively. Also let $\forall g \in G$, $\tau_{min} = min(d(g, g'; \{p, q\}))$, where $g, g' \in G$, $p \in P^{in}$, and $q \in P^{out}$. If $e^{ss}_{gmax}.t - e^{ss}_{gmin}.t < \tau_{min}$, then the partial snapshot is causally consistent.*

*Proof.* We extend the proof of Theorem 1 to a distributed setting. Similar to Theorem 1, there are three cases, with (3) being the one that differs. We again prove it by contradiction.

Assume $(e \in C_{part}) \wedge (\exists e' \to e)$ but $(e' \notin C_{part})$. As before, there must be some chain $e' \to e^{out} \to e^{g} \to e$. Because $e' \notin C_{part}$, we have $e^{ss}_{p^{in}_j} \to e'$ or $e^{ss}_{p^{in}_j} = e'$, that is, $p^{in}_j$ must have been triggered directly or indirectly by an inbound message. Denote the arrival of this inbound message at its marking gateway as $e^{g'}$. By the definition of $\tau_{min}$, we have $e^{g}.t - e^{g'}.t \geq \tau_{min} > e^{ss}_{gmax}.t - e^{ss}_{gmin}.t$. Thus, at event $e^{g}$, the gateway must have already initiated the snapshot and will mark $e^{g}.m$ before forwarding. This results in $e \notin C_{part}$, a contradiction! □

**Formal proof in paper**

**Theorem**: if ↔ < ↔, the partial snapshot is consistent!

↔ ≡ *Time gap between initiator-to-SLB one-way delays*

↔ ≡ *Time to form an external causal chain (GW→in-group→out-group→GW)*

***Observation: the condition holds in normal cases!***

↔ *can* ***approximate zero***

- *SLBs share the same region*
- *Proper placement of controller*

↔ *is relatively high*

- $\geq 3$ *trips through the fabric*
- *Higher when the out-group is in another DC or Internet*

**Theorem 2.** *In a system with multiple asynchronous gateways, let the wall-clock time of the first and last gateway snapshots be* $e^{ss}_{gmin} = \min_{e^{ss}_g}(e^{ss}_g.t)$ *and* $e^{ss}_{gmax} = \max_{e^{ss}_g}(e^{ss}_g.t)$*, respectively. Also let* $\forall g \in G$, $\tau_{min} = min(d(g, g'; \{p, q\}))$*, where* $g, g' \in G$, $p \in P^{in}$*, and* $q \in P^{out}$*. If* $e^{ss}_{gmax}.t - e^{ss}_{gmin}.t < \tau_{min}$*, then the partial snapshot is causally consistent.*

*Proof.* We extend the proof of Theorem 1 to a distributed setting. Similar to Theorem 1, there are three cases, with (3) being the one that differs. We again prove it by contradiction.

Assume $(e \in C_{part}) \wedge (\exists e' \rightarrow e)$ but $(e' \notin C_{part})$. As before, there must be some chain $e' \rightarrow e^{out} \rightarrow e^g \rightarrow e$. Because $e' \notin C_{part}$, we have $e^{ss}_{p^{in}_j} \rightarrow e'$ or $e^{ss}_{p^{in}_j} = e'$, that is, $p^{in}_j$ must have been triggered directly or indirectly by an inbound message. Denote the arrival of this inbound message at its marking gateway as $e^{g'}$. By the definition of $\tau_{min}$, we have $e^g.t - e^{g'}.t \geq \tau_{min} > e^{ss}_{gmax}.t - e^{ss}_{gmin}.t$. Thus, at event $e^g$, the gateway must have already initiated the snapshot and will mark $e^g.m$ before forwarding. This results in $e \notin C_{part}$, a contradiction! □

***Formal proof in paper***

***Optimistic Gateway Marking (OGM)***

*Optimistic execution in common cases*

*Verification/rejection of snapshots under worst cases*

# How does Beaver detect a snapshot violation?

**Theorem**: if ↔ < ↔, the partial snapshot is consistent

↔ ≡ *Time gap between initiator-to-SLB one-way delays*
↔ ≡ *Time to form an external causal chain (GW→in-group→out-group→GW)*

*1. Determine the lower bound of ↔ statically*

*2. Measure a safe upper bound for ↔ online using a single clock*

***False positives is fine as one can always retry!***

# Key ideas in Beaver 

*How to ensure consistency without coordinating external machines?*

**Idea 1: Indirection through Monolithic Gateway Marking**

*How to instantiate the theoretical model in practice?*

**Challenge 1** How to practically instantiate GW?

**Idea 2: Exploit the unique location of existing SLBs**

**Challenge 2** How to handle asynchronous GWs?

**Idea 3: Optimistic Gateway Marking (OGM)**

- Optimistic execution *in common cases*
- Verification/rejection of snapshot *under worst cases*

# Key ideas in Beaver

*How to ensure consistency without coordinating external machines?*

Ide

Ch

Ide

## More details about Beaver's protocol…

- Synchronization-free snapshot verification
- Supporting parallel snapshots
- Handling failures
- Handling packet loss, delay, and reordering
- …

**Challenge 2** How to handle asynchronous GWs?

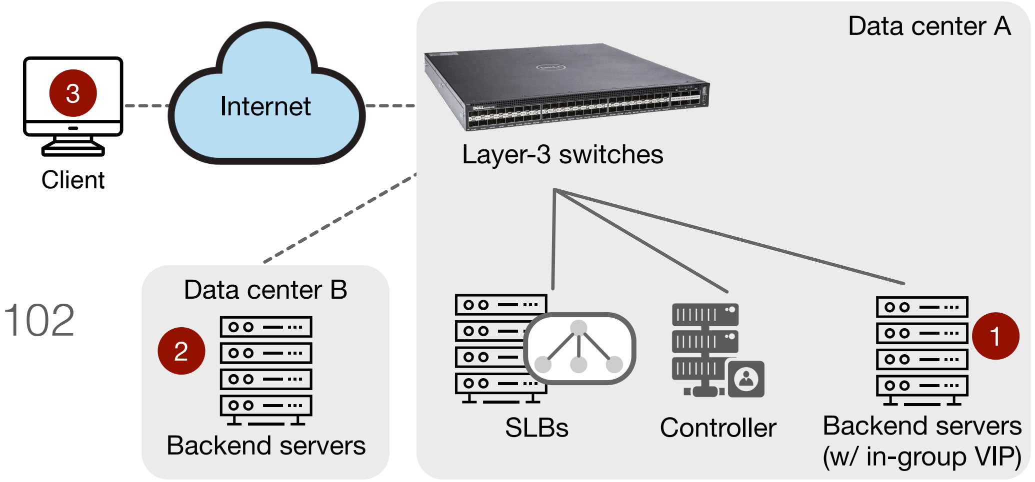
**Idea 3: Optimistic Gateway Marking (OGM)**

- Optimistic execution *in common cases*
- Verification/rejection of snapshot *under worst cases*

# Implementation and evaluation

## SLB-associated workflow

- Layer-3 ECMP forwarding per service VIPs: DELL EMC PowerSwitch S4048-ON
- Core SLB functions in DPDK: ~1860 LoC
- Backend server functions in XDP and tc: ~1040 LoC



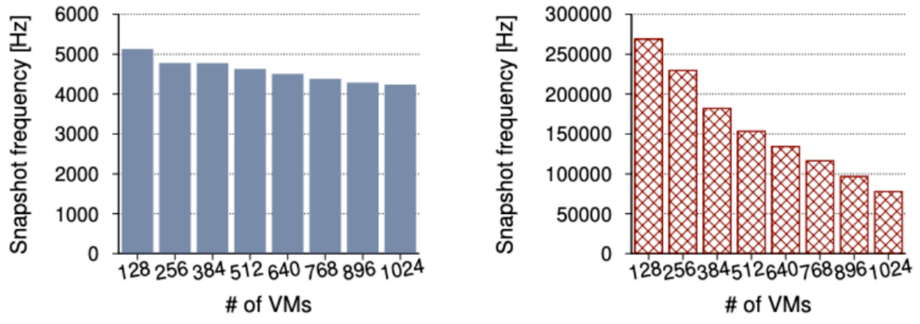

## Beaver protocol integration

- Minimal logic: (1) 68 LoC for SLB DPDK data path logic (2) 102 LoC for eBPF at in-group VMs

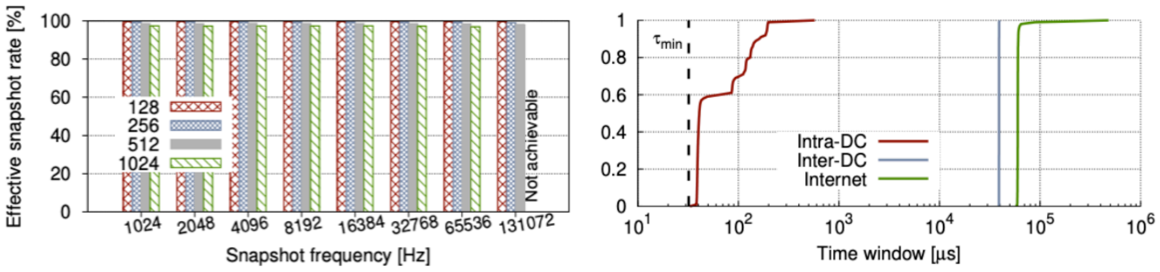
## Topology

- Support typical communication patterns
- Possible out-group locations: within the same DC, DC at a different region, or on the Internet
- Scale up to 16 SLB servers and 1024 backend applications

# Details in the paper…

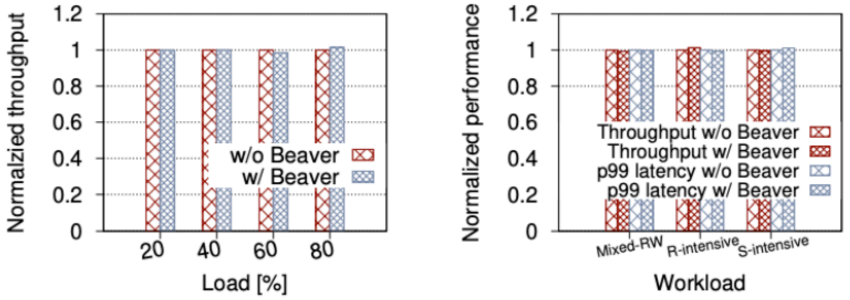



(a) w/o parallelism (b) w/ parallelism



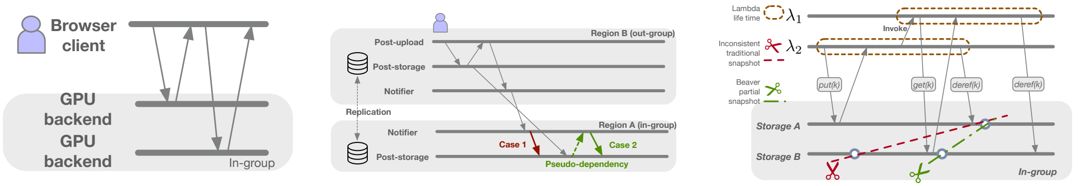

**Beaver supports fast snapshot rates**

**Beaver rejects snapshots infrequently**



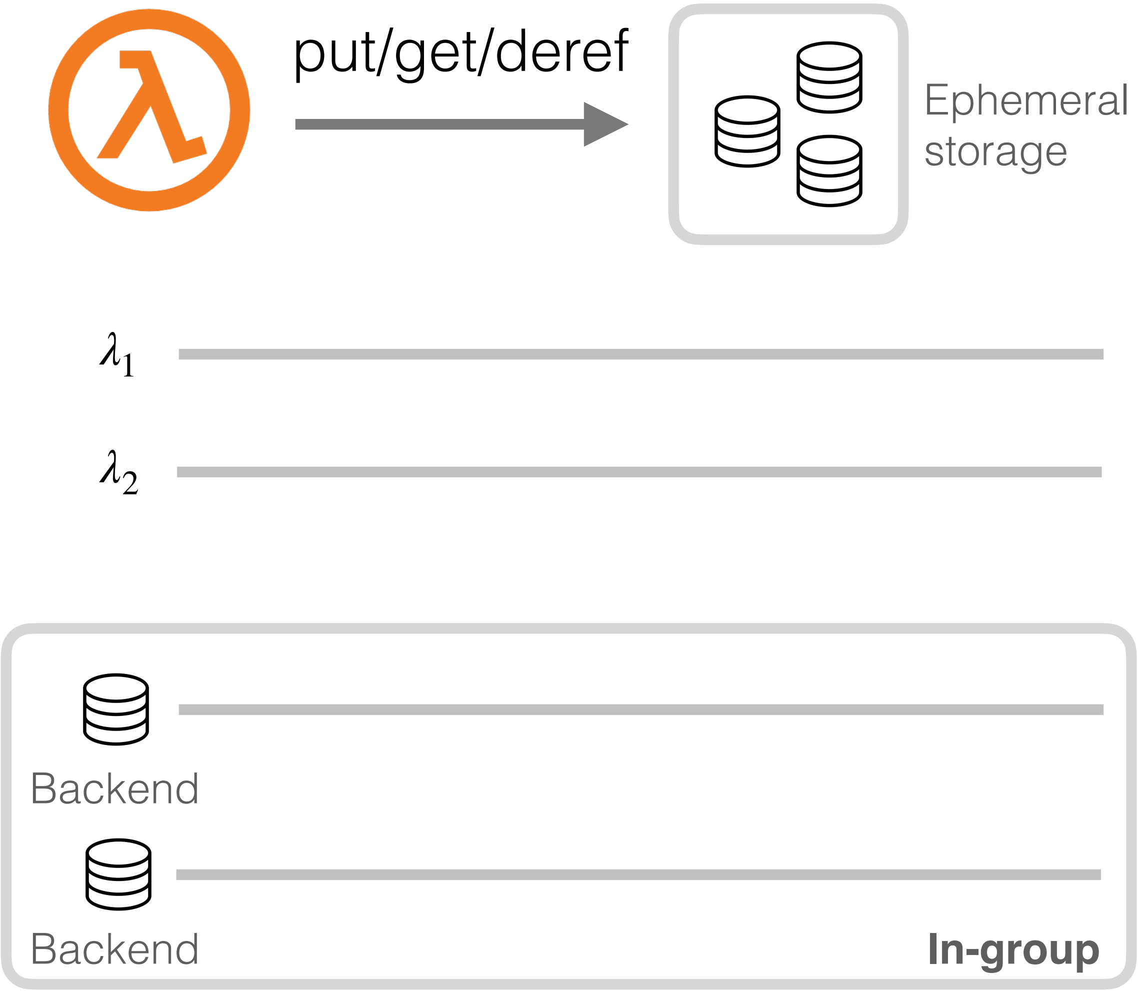

(a) Stressed workloads (b) YCSB benchmarks

**Beaver incurs zero impact**

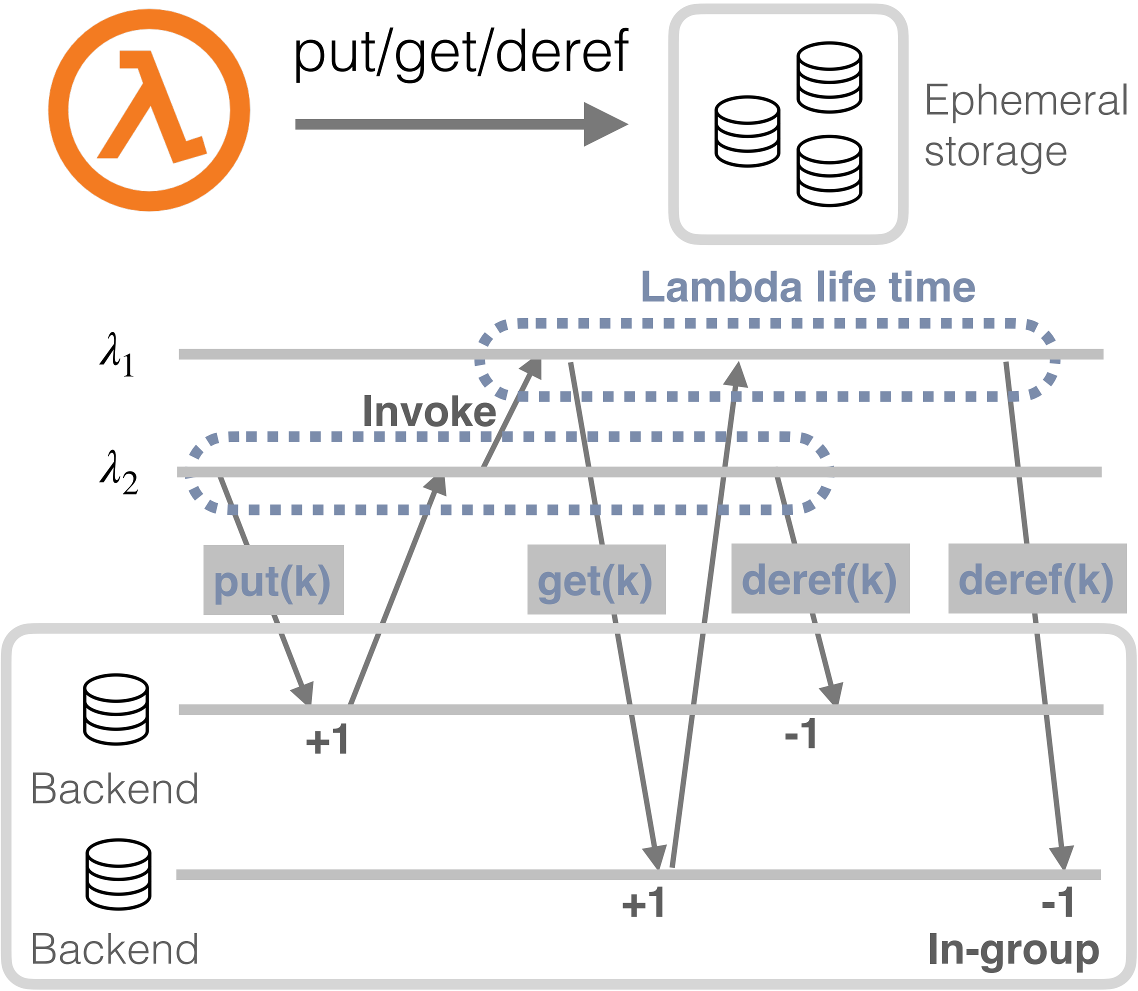



**Use cases: integration testing, service analytics, deadlock detection, garbage collection…**

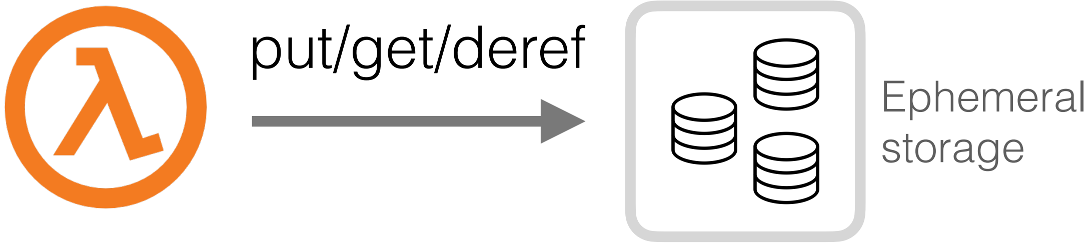
# Example: garbage collection for ephemeral storage

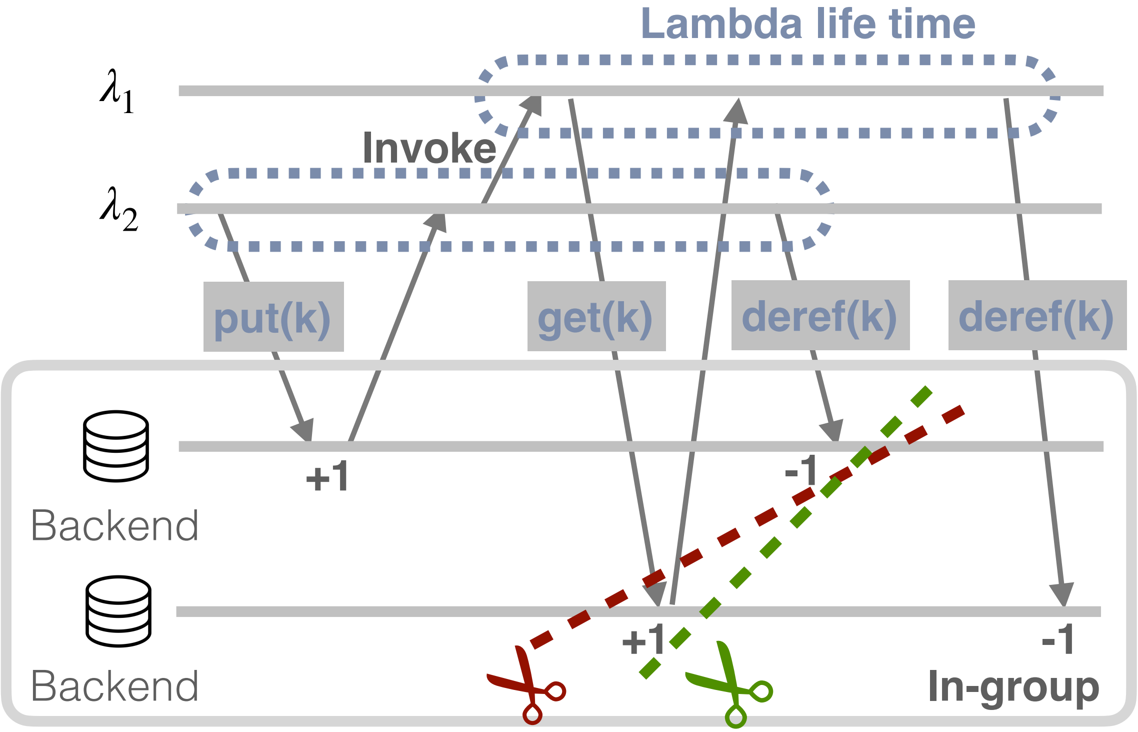


# Example: garbage collection for ephemeral storage



# Example: garbage collection for ephemeral storage



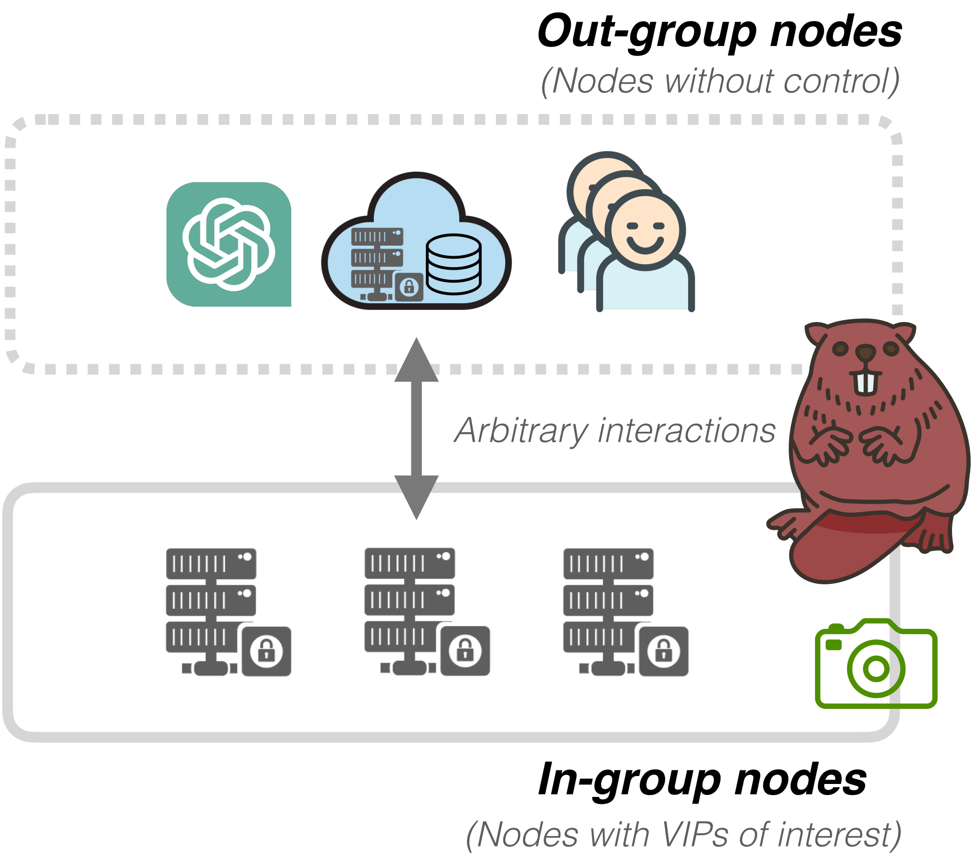




**Strawman**

Reference count = 0, unsafe recycle decision of $k$!

Reference count = 1, safe decision recognizing open reference to $k$

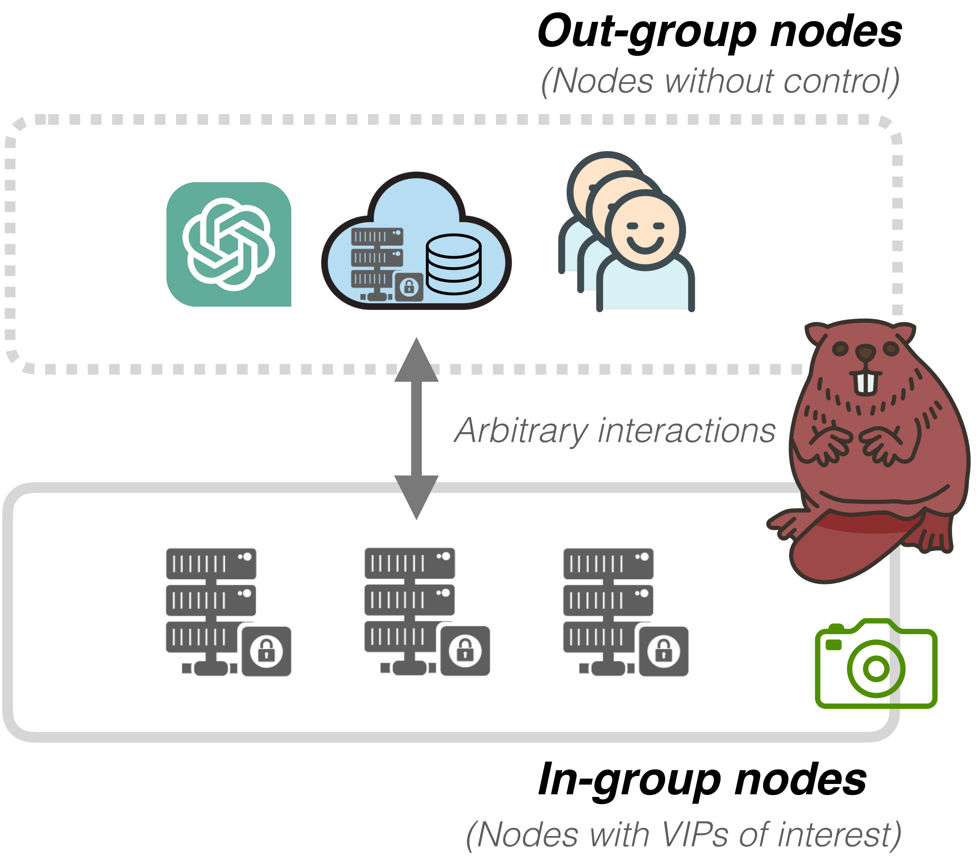
# Beaver: summary




The first practical partial snapshot protocol

- Extending classic distributed snapshot abstraction to partial deployment settings
- Incurring near-zero impact to existing traffic and minimal changes

# Beaver: summary




The first practical partial snapshot protocol

- Extending classic distributed snapshot abstraction to partial deployment settings
- Incurring near-zero impact to existing traffic and minimal changes

**Questions?**