Beaver:
Practical Partial Snapshots for Distributed Cloud Services

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

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

Unfortunately, the assumption mismatches the real-world scenarios!
The assumption mismatches the reality!

- Modular services
- Instrumentation constraints
- Costs and overheads
- Hidden causality due to human

Utopian: isolated ‘universe’ of nodes
The assumption mismatches **the reality!**

- **Unrealistic** to assume *zero* external interaction
- **Impractical** to instrument *all* processes

*Utopian: isolated ‘universe’ of nodes*

- Costs and overheads
- Hidden causality due to human
Consequences?

A single external node can break the guarantee!
Can we capture a **causally consistent** snapshot when a **subset** of the broader system participates?
Beaver: practical partial snapshots

**Out-group nodes**
(Nodes without control)

**In-group nodes**
(Nodes with VIPs of interest)

- 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

How is it even possible \textit{without} coordinating machines external to those of interest?

\textbf{Build a dam like a Beaver!}

\begin{itemize}
  \item \textbf{Out-group nodes} (Nodes without control)
  \item \textbf{In-group nodes} (Nodes with VIPs of interest)
  \item Zero impact over existing service traffic
  \item Absence of blocking nor any form of delaying operations.
\end{itemize}
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 （after $e_2$）

With GW: *consistent* cut at （before $e_2$）
Formalizing idea 1: Monolithic Gateway Marking

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

**Proof.** Let $e \cdot p = p^m_1$ and $e' \cdot p = p^m_2$. 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 $p \in P^m$.

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 $(e \in C_{\text{part}}) \land (\exists e' \rightarrow e)$ but $(e' \notin C_{\text{part}})$. With (3), $e' \rightarrow e$ means that there must exist some $e'' \in P^m$ (at an out-group process) satisfying $e' \rightarrow e'' \rightarrow e$. Now, because $e' \notin C_{\text{part}}$, we know $e'' \rightarrow e'$ or $e'' \rightarrow e'$, that is, $p^m_1$’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 $e'' \rightarrow e' \rightarrow e'' \rightarrow e$.

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

This contradicts our original assumption that $e \in C_{\text{part}}$. \qed

- 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**

\[ e_2 \text{ in snapshot, yet } e_0 \text{ that leads to it is not, inconsistent!} \]
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 $\leftrightarrow < \leftrightarrow$, the partial snapshot is consistent!

Intuition: the resulting snapshot is consistent

1. if $\leftrightarrow$ is large enough
2. or if $\leftrightarrow$ is ‘close’ enough

Observation: causally relevant messages are rare!

$\overline{\text{Challenge 2: handling multiple SLBs}}$

Reflection: beyond worst cases, how often does the violation occur?

Formal proof in paper

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 $(g \in \text{GW}_{in-group}) \land (g' \in \text{GW}_{out-group}) \land (g \neq g')$. As before, there must be some chain $e \to e'' \to e' \to e$. Because $e' \notin \text{GW}$, we have $e'' \to e' \notin \text{GW}_{in-group}$, which is, $\exists e''$ must have been triggered directly or indirectly by an inbound message. Denote the arrival of this inbound message at its marking gateway as $e'$. By the definition of $\text{GW}_{in-group}$, we have $e' \in \text{GW}_{in-group}$. Thus, at event $e'$, the gateway must have already initiated the snapshot and will mark $e''$ as before forwarding. This results in $e' \notin \text{GW}_{in-group}$, a contradiction!

Theorem 2. In a system with multiple asynchronous gateways, let the wall-clock time of the first and last gateway snapshots be $t_{GW}^{\text{in-group}}$ and $t_{GW}^{\text{out-group}}$, respectively. Also let $\forall g' \in \text{GW}_{in-group}, \exists g \in \text{GW}_{out-group}$, where \( g', g \in G, p \in P \), and $q \in P'$. If $t_{GW}^{\text{in-group}} < t_{GW}^{\text{out-group}}$, then the partial snapshot is causally consistent.
Theorem: if $\leftrightarrow < \leftrightarrow$, the partial snapshot is consistent!

\[ \equiv \text{Time gap between initiator-to-SLB one-way delays} \]
\[ \equiv \text{Time to form an external causal chain (GW→in-group→out-group→GW)} \]

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 $\epsilon_{\text{first}} = \min_P \{ e_i \}$ and $\epsilon_{\text{last}} = \max_P \{ e'_i \}$, respectively. Also let $y \in G, x_{\text{in}} = \min_P \{ d_i \}$, where $y \leq x_{\text{in}}$ and $x_{\text{in}} \leq y_{\text{out}}$. If $\epsilon_{\text{last}} - \epsilon_{\text{first}} < \epsilon_{\text{out}}$, 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 $(e \in \epsilon_{\text{last}}) \land ((e' \rightarrow e) \land (e' \in \epsilon_{\text{out}}))$. As before, there must be some chain $e \rightarrow e'' \rightarrow e' \rightarrow e$. Because $e \notin \epsilon_{\text{out}}$, we have $e'' \neq e'$ or $e'' = e'$, that is, $e''$ must have been triggered directly or indirectly by an inbound message. Denote the arrival of this inbound message at its marking gateway as $e''$. By the definition of $\epsilon_{\text{out}}$, we have $e'' \rightarrow e'$. Thus, at event $e'$, the gateway must have already initiated the snapshot and will mark $e'$ as before forwarding. This results in $e \notin \epsilon_{\text{out}}$, 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 $\leftarrow \leftarrow < \leftarrow \leftarrow$, the partial snapshot is consistent

$\leftarrow \leftarrow \equiv$ Time gap between initiator-to-SLB one-way delays

$\leftarrow \leftarrow \equiv$ Time to form an external causal chain (GW $\rightarrow$ in-group $\rightarrow$ out-group $\rightarrow$ GW)

1. Determine the lower bound of $\leftarrow \leftarrow$ statically
2. Measure a safe upper bound for $\leftarrow \leftarrow$ 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?

Idea 1: Indirection through Monolithic Gateway Marking

Challenge 1: How to practically instantiate GW?

Challenge 2: How to handle asynchronous GWs?

Idea 2: Exploit the unique location of existing SLBs

Idea 3: Optimistic Gateway Marking (OGM)

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

More details about Beaver’s protocol...

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

Beaver supports fast snapshot rates

Beaver incurs zero impact

Beaver rejects snapshots infrequently

Use cases: integration testing, service analytics, deadlock detection, garbage collection…
Example: garbage collection for ephemeral storage

\[ \lambda_1 \]

\[ \lambda_2 \]

put/get/deref

Ephemeral storage

Backend

In-group
Example: garbage collection for ephemeral storage

Lambda life time

put(k)
get(k)
deref(k)
deref(k)

put/get/deref

Ephemeral storage

Backend

+1 +1

-1 -1

In-group

Backend

λ₁

λ₂

Invoke

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
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**

- **Out-group nodes** (Nodes without control)
- **In-group nodes** (Nodes with VIPs of interest)

Questions?