Remus: High Availability via Asynchronous Virtual Machine Replication

Key Takeaways

- Remus modifies the virtual machine to provide high availability, and this can work in cases where the operating system source code is not available.
- The configuration consists of a primary machine and a backup machine. Periodically, the state of primary machine is fully replicated and transmitted to the backup machine, and the output of primary machine is released to client only after the acknowledgement of application of change from the backup machine.
- The frequent replication of full system cause significant delay in the output, and can make the throughput as low as 1/4 compared with original unprotected virtual machine.

High Level Properties

- generality: the high availability can be provided as low-level service, applicable regardless of application being protected or hardward it runs on.
- transparency: the high availability can be provided without modifying the operating system which might not even have source code available
- seamless failure recovery: no externally visible state should ever be lost, and failure recovery should proceed rapidly enough that it appears as nothing more than temporary packet loss

Approach

- The system is based on a modified version of virtual machine.
- There are two machines running, one primary and one backup.
- The state of the primary machine is replicated and transmitted to the backup machine periodically. The backup machine applies the transmitted image of the primary machine to its state then sends the complete signal back to the primary machine.
- The primary holds the message requested from the user until it receives the message from the backup machine that the replication has been applied to the backup machine, so the state is visible from outside only after it has been backuped.
- The backup is performed periodically, hence all changes to the primary machine is buffered in memory before the checkpoint timestamp.
- When the backup image is transmitted to the backup machine but the primary machine hasn't received complete message, the primary machine still performs computation during this interval and this is called speculative execution.
- The backup frequency is about 40 times per second.
- In addition to memory and CPU state, the disk state between primary and backup machine also needs to be consistent. Therefore, if a write happens on primary machine, the write is transmitted to the backup machine, and the backup machine holds the write in memory buffer, then actually writes to the disk when the checkpoint message from the primary machine comes.
- The virtual machine does not actually run on the backup machine. Instead, it only tries to keep a state. Hence the backup only consumes small resouce, and one backup machine can be used for backup of multiple primary machines.
- When the primary machine tries to capture all changes, it pauses its own operation and copies all changes to a separate location. Then the primary machine resumes operation and also sends the image to the backup machine.

Detecting Failure

- A heartbeat sent between primary and backup machines is used to detect failure.

- If the backup does not receive the heatbeat for a while, it assumes the primary fails and take over from the latest checkpoint.
- If the primary does not receive the heartbeat for a while, it assumes the backup fails and disables protection.
- In the experiment, the primary and backup machines are connected by a pair of bonded network interface using Ethernet crossover tables, hence a primary/backup machine pair is two colocated machines.

Evaluation

- The system is tested by injecting network failures at each phase of the replication protocal, while putting substantial disk, network and CPU load on the protected system. The result shows at each stage of primary machine failure, the backup machine successfully takes over, and the backup machine has a consistent disk with the primary machine disk.
- The recovery takes about 1 second when the primary machine fails.
- The replication process puts some burden on the machine. A general purpose task such as kernel compilation incurs approximately 50% penalty, and the network dependent workloads represented by SPECweb benchmark gets speed of only 1/4 of original. Therefore, Remus still causes significant network delay particularly for those exhibiting poor memory locality, i.e. when many page tables are modified at each interval.
- The time spent copying memory state is a lot less compared with the time spent transmitting copied state to the backup machine, and the time spent is linear to the frequency of checkpoint.
- The SPECweb benchmark is dominated by the delay caused by holding the output of primary machine until the checkpoint is completed.
- The Postmark disk benchmark shows the replication has no significant impact on disk performance.

Potential Improvement

- The replication transmission can be improved by compressing the transmitted information. The message can contain only the change of page instead of full page information, or get compressed using a general purpose compression algorithm. Since the memory replication takes a lot less time compared with image transmission, it seems plausible to compress the image before actually sending it to the backup machine.