PARALLEL PROCESSING WITH PERSISTENT MEMORY CLONES

pivotnine

SPONSORED BY: MemVerge

Author: Justin Warren

April 2021

WP

THE POWER OF MEMORY CLONES

Using MemVerge's memory snapshots and clones allows parallel use of in-memory datasets in ways that are difficult, if not impossible, using traditional techniques.

Memory clones open up a range of interesting workload consolidation opportunities that provide substantial economic and efficiency benefits compared to more complex, distributed scaling methods. By sharing resources, workloads can use the same in-memory dataset in parallel, quickly gaining access to the latest data from earlier processing stages.

Creative and exploratory work can benefit from working in parallel from in-memory clones of data. Performing *what-if* analysis from a common baseline allows exploration of different ideas from the same shared starting point.

By providing quick restore points for rapid recovery of large datasets, exploration of different creative pathways can be done with low risk. Novel ideas can be investigated with confidence that a misstep will not require hours of reconstructive work to return to a checkpoint. Program crashes no longer derail a team's ability to move forward due to the failure of a single, critical link in the chain.

Multi-Tenant Memory

Taking copies of the same in-memory program state and making them available to multiple people provides a level of parallelism that we are familiar with in other contexts, but have been unable to use with memory state. Until now.

Snapshots and clones of in-memory datasets permit virtualization of memory in a similar way to how VMware provides shared access to compute and storage infrastructure. Shared access to memory has, so far, been limited to traditional operating system virtual memory architectures that impose a heavy penalty if the active working set is very large.

Thinly-provisioned storage is possible partly due to the use of copy-on-write storage systems, and MemVerge's Memory Machine architecture uses a similar approach to provide memory snapshots and clones. Just as VMware provided a compatibility layer between existing programs and physical infrastructure, MemVerge virtualizes physical DRAM and persistent memory to provide a large pool of memory resources in a way existing programs can easily consume.

Keeping data in memory makes fast access to even very large datasets possible. Programs can be brought to the data, rather than having to replicate datasets to where the programs are. Like the early days of VMware, this provides opportunities for workload consolidation to maximize the use of relatively expensive infrastructure.

Rather than purchasing DRAM for many nodes at great expense and using complex scale-out data processing methods, fewer, larger nodes can be used. Expensive DRAM can be augmented with relatively cheap PMEM to provide very large per-node memory pools. By using MemVerge's Memory Machine approach, datasets can be reused with only changes consuming additional memory, increasing efficiency.

A single, primary dataset can be maintained as a kind of 'golden image' that is cloned for use by a range of workloads without requiring each workload to take a full copy of the original data. For poly-dense animation scenes or multi-terabyte analytics datasets, this can add up to substantial savings for read-heavy data access.

WP

Reporting and Analysis

Creating a reporting copy of a production database is a common design pattern where long-running, resource intensive reporting jobs could interfere with an online, transaction processing environment. The reporting copy provides a point-in-time clone of a dataset that is suitable for summary reporting that doesn't need the very latest transactions to be useful.

This design pattern was particularly common in the pre-flash disk storage era, where contention for limited disk I/O could severely impact the performance of latency-sensitive applications. The challenge is still present in all-flash systems, though the more consistent latency and throughput of flash compared to spinning-disk media has dramatically improved the situation.

However, for workloads that use very large in-memory datasets—such as AI/ML applications, animation, genomics, and other analytics workloads—the temporary pause in processing to flush a consistent copy of state to storage media can be unacceptable. Modern analytics datasets can be many terabytes in size, which take minutes to flush to storage even with very fast all-flash storage arrays and high-bandwidth links. A multi-minute pause in transaction processing is functionally equivalent to an outage for these systems, and regularly scheduled outages are rightly seen as a legacy approach from last century.

By using a snapshot of the memory state with something like MemVerge's memory machine, the pause in activity can be reduced dramatically. A snapshot can be taken in less than a second, a small enough window that production activities aren't impacted unduly. And as the snapshot data is contained in memory already, a clone can be created—still in memory—without waiting for data to load from relatively slow storage media. The clone can then be used for parallel activities while the original processing thread continues uninterrupted.

Refreshing the data is fast and straightforward, removing the overhead of long flush and reload cycles. This can speed up end-to-end processing where multiple stages arranged in series are required, but the parallel processing potential is what makes in-memory dataset clones particularly interesting.

Like VMware, the benefits of software-defined-infastructure extend into new services enabled by virtualization. Co-locating processing of large in-memory datasets unlocks a host of new data processing approaches that are API-driven and highly automated.

WP

Genomic Sequencing and Analytics

Single Cell RNA (scRNA) sequencing analysis uses a computational model with large datasets, and a multi-stage pipeline with multiple intermediate stages. As the data is analyzed, branching off these intermediate stages is used to perform *what-if* analyses.

Due to the size of the datasets[1], flushing the memory state to disk and then re-importing the data from files back into memory for processing can take over 15 minutes for each stage. This affects the design of the pipeline stages and the types of analysis that can be undertaken.

In one MemVerge customer, an 11 stage pipeline was only able to complete an average of three analysis tasks per hour due to I/O overheads of saving and restoring memory state consuming 61% of task completion time. By using in-memory snapshots, I/O overhead was reduced to a mere 3% of task time and the customer was able to move to a multi-branch pipeline where multiple branches of analysis could be performed in parallel.

Keeping Animation Moving

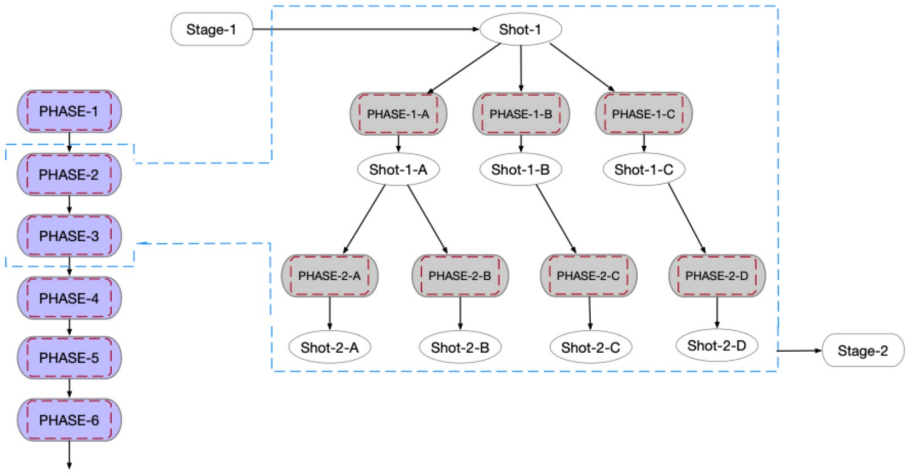
Polygon dense scenes in industry standard applications like Autodesk Maya 3D graphics software can take minutes to save and load with even fast flash storage. Minutes-long save times discourage artists from saving their work regularly, even though a crash can mean half a day of rework will be needed.

The collaborative nature of artistic work means that one crash and the resultant delay can mean dozens of other artists are also delayed waiting on the work to be redone. Production timelines are often tight, and even a half day delay could cost a major production more than the cost of a few persistent memory DIMMs.

A memory snapshot that completes in under a second removes the disincentive for artists to regularly checkpoint their work, and keeps the entire system moving.

By rapidly restoring from snapshots, artists can drop back into a full Maya session exactly configured to where they left off. They can test ideas, safe in the knowledge that abandoning something that didn't work won't result in waiting an hour for a scene to load back in from storage.

Multiple artists can work in parallel from the same baseline snapshot, trying out alternatives before deciding on a final approach to take. Instead of hampering their



Parallel task execution in substages using memory clones.

[1]For example, the Mouse Cell Atlas (GSE108097)

creative efforts, fast snapshots can help them to take creative risks without endangering production deadlines, resulting in a higher quality end result.

Development and Debugging

Live memory clones are very useful for software developers hunting for tricky bugs, particularly the kind of bugs that emerge at runtime in complex systems.

Running in production with full debug enabled has unacceptable performance implications, and there are also dangers in turning on live instrumentation of a running production system. Monitoring systems such as distributed tracing are also limited to systems that have implemented tracing extensively throughout the code, and have appropriate processes for using the data.

When things go wrong, there's nothing quite like being able to see the broken system in its natural environment.

Taking a copy of the program while it is experiencing a problem means it can be examined carefully, either in situ on the production infrastructure or in a safer, non-production environment in order to locate the source of the problem without endangering the running program or any production data. Taking copies of the running, production state can allow multiple people to all look at the same system in parallel, altering its state as they investigate their own hypotheses.

Some bugs don't manifest until after substantial runtime, such as certain kinds of memory leaks or interactions with other long-running components. Until the cause is located, it can be difficult to replicate the conditions leading to the bug. By taking a snapshot of the running program after it has reached the error condition, it can be examined, potentially repeatedly, without requiring a restart and warmup time for each test run.

Again, multiple developers can all access clones of the same primary memory snapshot and start from the same point, exploring on their own in parallel to one another. Just as storage snapshots provide rapid access to baseline datasets, memory snapshots can provide rapid access to large in-memory datasets without incurring a setup penalty.

When things go wrong, there's nothing quite like being able to see the broken system in its natural environment.

This technique isn't limited to production systems. Forensic analysis of running programs during development can quickly resolve issues that take much longer to find and fix using more traditional debugging techniques. Intermittent errors, rare dependency conditions, or challenging asynchronous code can all benefit from live inspection rather than painstakingly searching for just the right breakpoints or error logging to finally locate the errant line of code. This can be particularly beneficial when combined with a reduction in load time for large datasets using memory snapshots instead of loading from storage.

WP

Security Forensics

The ability to snapshot and clone live memory state provides some intriguing possibilities for security forensics.

Security researchers make extensive use of virtual machine sandboxes to safely isolate dangerous code, so they are already familiar with the snapshot and cloning process. However, the lengthy time it takes to stun and store a virtual machine image can make exploratory work tedious and difficult. Resetting a virtual machine environment to a known state before injecting malware for analysis increases cycle times substantially.

A live process snapshot that can be cloned, or copied to another machine, and restored to its original state time and time again could allow detailed state examination that isn't possible or straightforward with existing tools. Providing multiple researchers and analysts parallel access to the same live malware sample will accelerate analysis. When coupled with the fast-restart feature of memory snapshots, iterative analysis on live program state becomes a feasible and vastly less tedious approach than the relatively slow process of restoring virtual machine state from storage.

Memory snapshots can be taken of both malware processes and entire virtual machines, depending on the granularity that is required, and with little to no impact on snapshot and restore times thanks to the copy-on-write architecture of MemVerge's memory machine.

While it is difficult to predict what new techniques will be developed, we are keen to see what the creative and non-linear minds of information security come up with.

WP

Conclusion

Memory snapshots provide a handy hybrid of familiar techniques and intriguing new ones.

We can use the knowledge we've already accumulated from decades of experience with storage snapshots and clones to make immediate use of memory snapshots in clearly beneficial ways. Rapid restore points for large datasets that can be easily cloned and worked on in parallel is a well-known technique with clearly understood benefits. For certain workloads, and at the right price-point, a memory-speed version of storage snapshots is trivially easy to justify.

What we find most exciting about memory snapshots is the potential for new techniques that have not yet been developed.

Completely new ideas are difficult to grapple with. They tend to require such a radical rethink of how we work that it's hard to know where to begin... and so we don't. Memory snapshots provide a helpful bridge between what we already know and the beguiling possibilities of the unknown. We are not required to leap into the void without a safety net to catch us if something we try doesn't work out.

This buys us time. Time to gain familiarity with how memory snapshots work, and how they are both similar and different to what we already know. That time isn't wasted, because we know we'll be making things better from the outset.

What we don't know is how memory snapshots might allow us to make radical, more fundamental changes to how we work. Teleporting a process from one node in a cluster to another node, complete with all of its memory state, isn't something we really do today. The ability to try it out might give rise to entirely new ways of managing cluster rebuilds or autoscaling.

PN

About The Author

Justin Warren is Founder and Chief Analyst of PivotNine. He has worked with many well-known companies around the world, including ANZ, Australia Post, IBM, NetApp, Nutanix, Pure Storage, Red Hat, Suncorp, Telstra, and VMware as well as a variety of startups including Atomist, CodeSee, Cloudistics, Datera, Elastifile, Env0, Habrdata, Hasura, Illumio, Isovalent, Manifold, Mattermost, Smallstep, Spectro Cloud, and Solo.io, among others.

Justin has written for a variety of mastheads, including *Forbes.com*, *iTnews*, *CRN Australia* and *The Saturday Paper*.

Justin holds an MBA from Melbourne Business School, and is a graduate member of the Australian Institute of Company Directors.

About PivotNine

PivotNine Pty Ltd is a specialist IT consulting firm based in Melbourne, Australia.

PivotNine helps customers to evaluate and select technology products, and to implement effective organisational structures and processes.

PivotNine assists vendors with marketing positioning and messaging, with a focus on data driven marketing methods.

Contact

enquiries@pivotnine.com

https://pivotnine.com