Providing Native Support for Byte-Addressable Persistent Memory in Golang

Pratap Subrahmanyam

VMware, Inc.

Use Cases

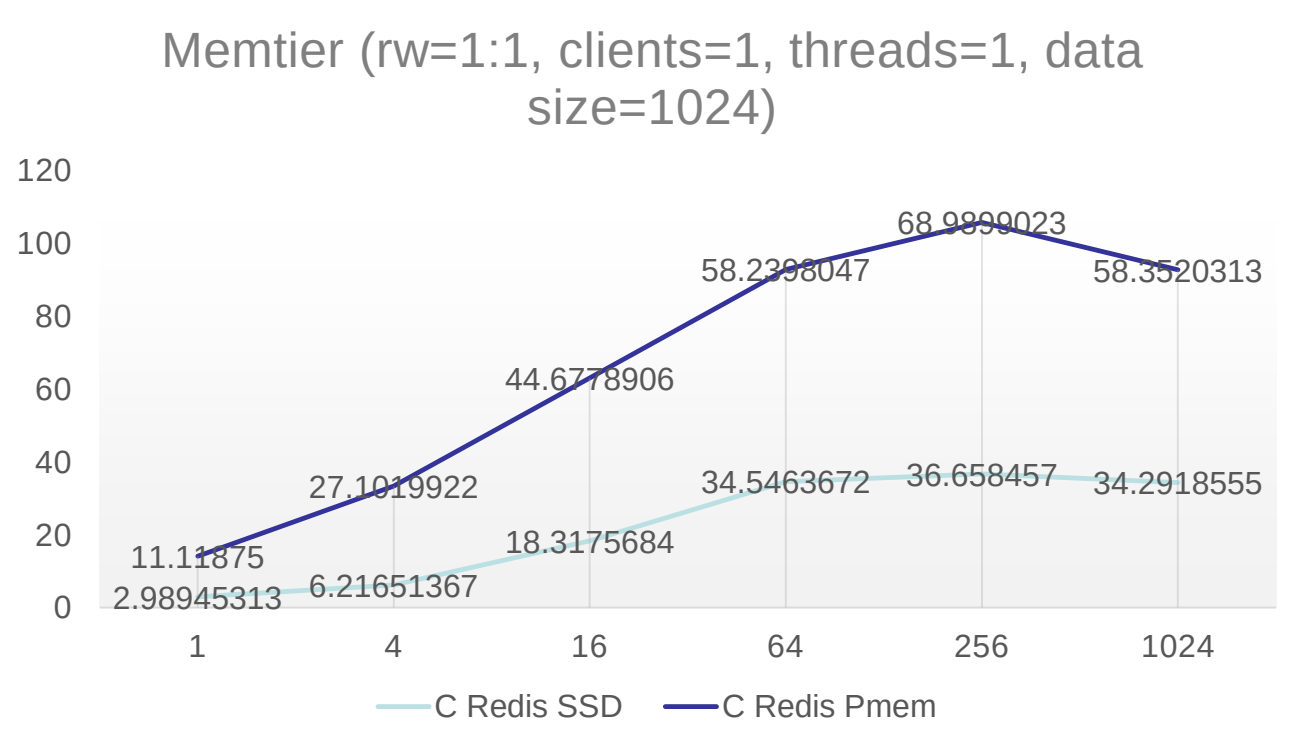
- Density - Augment fast, low density DRAM with
  - slower, high density 3DXpoint in non-persistent mode
  - Large graph processing applications
  - Avoid the management ugliness of scale out
- Tiered Storage - Augment SSDs with
  - expensive, low latency storage
- Persistence with Random Access - Use the load/store model
  - Large in-memory databases
    - SAP HANA, Redis, SQL …
  - Large AI models
    - Starting to appear in accelerator boards

What we did …

- First things first
  - Virtualized Persistent Memory
  - Zero day hardware support
- Avoided the ramp-up items, chose the hardest one to explore
  - The load/store model
- 3 Questions
  1. After all the logging, write amplification, cache flushes for crash consistency, is there still performance to be gained?
  2. Is the code readable?
     - Programming with Persistent Memory
     - Identifying transactions
  3. Coping with crash consistency, durability, availability

Flash Memory Summit

Performance Results



Gruesome Experience!

- Hand insertion of logs is a killer!
- Would be great to have persistent variables reside on the heap!

Our Goal …

```
// Adds a node to the linked list and updates the tail and head
func addNode(tx transaction.TX, rptr *root) {
	txn {
	entry := new(entry)
		entry := pnew(entry)
	entry.id = rand.Intn(100)
		entry.id = rand.Intn(100)

	if rptr.head == nil {
		if rptr.head == nil {
		rptr.head = entry
			rptr.head = entry
	} else {
		} else {
		rptr.tail.next = entry
			rptr.tail.next = entry
		}
	}
		rptr.tail = entry
	rptr.tail = entry
	}
}
```

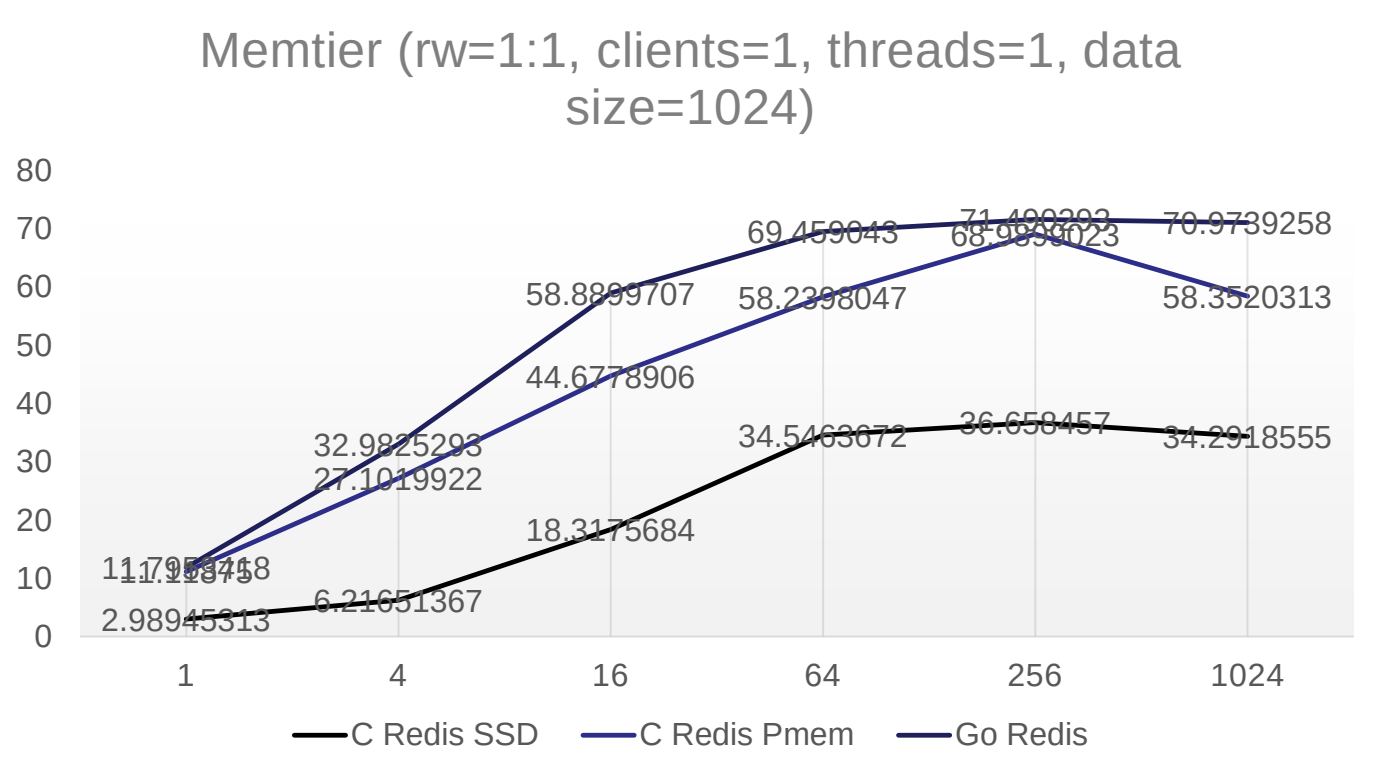
```
// The root object
type root struct {
	head  *entry
	tail  *entry
}

// Structure of each node in the linked
// list
type entry struct {
	id   int
	next *entry
}
```

Repositories

- Runtime changes to support persistent heaps: https://github.com/jerrinsg/go-pmem
- Libraries for transactions support, logging: https://github.com/vmware/go-pmem-transaction
- A partial Redis rewritten in Go: https://github.com/vmware-samples/go-redis-pmem

Performance



Conclusion

- All VMware VMs support virtualized persistent memory
- Language support is necessary!