Dollars for Disks

Thoughts on Running Postgresql in the Cloud Computing Era Jeffrey Zampieron – Postgres Conf NY 2019

About Me

- ► Technical Lead/Architect @ MedAcuity Software
 - MedTech Consulting
 - Specializing in software compliant w/ ISO13485:2016 and ISO/IEC62304
 - https://www.medacuitysoftware.com/
- Previous
 - CTO/Director of Cloud @ Beco/Convene Technology
 - ▶ IoT/Mobile/Cloud system for Experiential User Engagement
 - ▶ Big-Data for the CRE Stakeholder
 - Sr. Tech Staff @ Systems & Technology Research LLC
 - ▶ DoD/IC Research into UxV navigation, Vision, ML, Cyber
 - ▶ Hyper-scale data processing via private cloud systems

Motivation

- Where did this talk come from?
 - Lessons learned running PG in the cloud
 - ► A startup needing to be efficient to survive
- Goals
 - System engineering approach to PG solutions
 - Provoke thoughts: Data and Example to guide evaluating your own situation
- Non-goals
 - ► Exhaustive PG performance or cost studies
 - ▶ Tuning and tweaking every last ounce of \$/TPS out of PG

Background

- Problem Statement
 - High volume/High concurrency data ingest (OLTP)
 - Real Time and Historical Analytics (OLAP)
 - Management/Provisioning and Production Fulfillment (ERP)
- Assets & Constraints
 - Limited budget (cloud credits)
 - Small Dev/Ops team
 - ▶ No legacy concerns
- Time to market is critical
 - ▶ The first big customer just landed

Landing Pad – System Overview

- Cloud Native Architecture
 - Azure
 - ▶ DC/OS Mesos
 - Stateful
 - ▶ Postgres 9/10
 - Kafka
- Stateful on dedicated VMs
 - ▶ I/O contention on shared VMs was problematic
- Initial PG design focused only on required application performance
 - Expected PG load + headroom

Landing Pad - PG

- ▶ PG was running on a Standard D32s v3 \$2885.15 / month
 - ▶ 32 vCPU, 128G RAM w/ 128G AMD SSD \$1330.88/month (PAYG)
 - ▶ Data (4xP40 2TB RAIDO), Log (2xP40 2TB RAIDO)
 - ► 6xP40 @ \$259.05/month = \$1554.27/month
 - Prices as of 17-March-2019
- Also had a read-replica Duplicate Machine
- Spending ~\$5500/month just on Postgres
- Performance was ~6000TPS
- Observation: Disks and I/O in the cloud is expensive!
 - ▶ First thought is usually... what did I do wrong?

Dollars for Disks – The Wrong Way

Scenario

- ▶ At the time, we were on Azure and not moving.
- Spending lots of \$/month on disks
- Are we getting what we are paying for?
- ▶ What other options exist?
- ▶ What do we really need?
- How we landed in the above scenario is an interesting story itself...
- Focus in on "what do we need"
 - ▶ With some measure of "Are we getting what we are paying for?"
 - ▶ Plus reasonable room to grow

What do you need – System Eng.

- Looking at cost reducing the PG server (largest spend)
- The rough approach Trying "the right way"
 - Start with top-level customer visible requirements
 - Define some PG KPIs (derived requirements)
 - Survey the landscape
 - Identify limitations
 - Prototype and Test
 - Implement
 - ▶ Profit!?!?
- Foreshadowing: AWS tests results and thoughts

Top-Level Requirements

- Can be loosely defined as long as they are bounded
- Scenario below is what we used to inform the tests
- ► SLAs
 - ▶ Uptime (planned/unplanned) 99%, ~3 days per year
 - ▶ RTO 8 hours
 - ▶ RPO 1 day
- Response Time Human perceptible (less than 250ms 99%)
 - Context dependent
- Concurrent Users/Application TPS Load
 - ► ~500k concurrent handsets application specific

PG KPIs

- ▶ Based on the previous slide we derive the following PG goals:
 - ▶ Replica is a nice-to-have. Manual fail-over is tolerable in minutes
 - ▶ 5000TPS min
 - Daily backups
- What's the lowest TCO way to achieve the above?
- TCO considerations
 - ▶ NRE
 - ► PG Deployment setup
 - Backup/Restore/Failover tooling
 - Ops
 - ▶ Recurring laaS/DBaaS depending on solution

Survey the Landscape

- We selected the following landscape to limit test scope and cost
- DBaaS
 - ► AWS RDS
- Self-Hosted
 - ► AWS EC2
- Metal
 - Private "Datacenter"

Bounds – Identified Limitations

- ▶ RDS
 - ▶ RAM, CPU, IOPS, I/O Throughput, Network Throughput
- ▶ EC2
 - ▶ RAM, CPU, IOPS, I/O Throughput, Disks per VM, Network Throughput
 - ► EBS
 - ▶ IOPS per Vol, Capacity per Vol
- Metal
 - ▶ RAM, CPU, IOPS, I/O Throughput, Disks per Box, Network Throughput

Bounds - Caveats

- When is persistent storage not actually persistent?
 - When its an AWS Instance Store.
 - ▶ When its an Azure "Temporary" Disk/Cache
 - ▶ Thought exercise: How are these super-fast, low-latency disk usable for a database? Is this a problem on metal?
- ► FMEA
 - What does AWS/Azure/etc talk about as far as failure modes?
 - What are the effects and recovery scenarios?
 - ► How much redundancy do I need in my system for an application SLA?

Prototype and Test

- All tests are run on AWS or Metal
- Using the same tooling
- IaaC, Tooling and Raw/Collected Results all provided
 - https://github.com/jzampieron/pgconf2019
 - Master branch is the "release" branch
 - Develop branch is my work in progress

Results

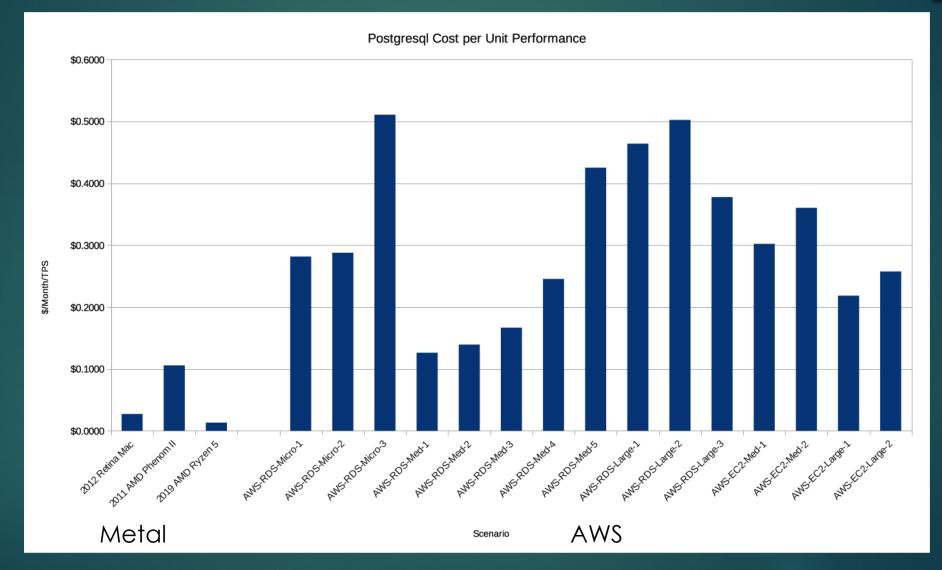
- \$/month/TPS are used to normalize cost
- Recurring costs are included in the \$/month/TP\$
 - ▶ Power, Cooling, Network, etc
 - Assumptions about costs are listed and are rough estimates
 - ▶ Use your own numbers if repeating this exercise
- ▶ Non-recurring costs
 - Amortized over 36 months for cap-ex into \$/month
 - ▶ 36 months is a typical depreciation schedule for cap-ex assets.
 - Amortized over 18 months for NRE into \$/month
 - ▶ 18 months is the expected design life for a cloud implementation
 - ▶ Your design lifespan may be different. Adjust accordingly.

Results Matrix

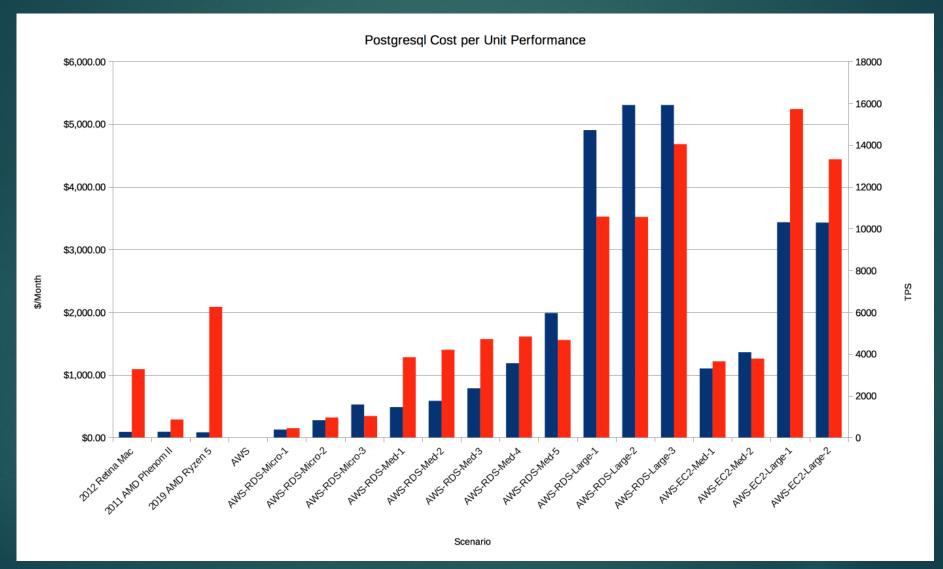
PG Costing	_					
Test Scenario	Notes	Cost / Month	# Client Machines	TPS/Client Machine	TPS	\$/month/TPS
Metal	Notes	Cost/Month	# Chefft Machines	(avy)	173	\$/IIIOIIII/1P3
2012 Retina Mac	apfs	\$88.92	1	3271	3271	\$0.0272
2011 AMD Phenom II	SSD Run	\$90.94	1	i	861	\$0.1056
2019 AMD Ryzen 5	btrfs RAID1	\$82.61	1			
AWS	,	,				
AWS-RDS-Micro-1		\$125.64	1	446	446	\$0.2817
AWS-RDS-Micro-2		\$275.64	1	958	958	\$0.2877
AWS-RDS-Micro-3		\$525.64	1	1029	1029	\$0.5108
AWS-RDS-Med-1		\$484.88	1	3843	3843	\$0.1262
AWS-RDS-Med-2		\$584.88	1	4199	4199	\$0.1393
AWS-RDS-Med-3		\$784.88	1		4709	\$0.1667
AWS-RDS-Med-4		\$1,184.88	1	4829	4829	\$0.2454
AWS-RDS-Med-5		\$1,984.88	1	4668	4668	\$0.4252
AWS-RDS-Large-1		\$4,905.00	1	10572	10572	\$0.4640
AWS-RDS-Large-2		\$5,305.00	1	10559	10559	\$0.5024
AWS-RDS-Large-3		\$5,305.00	2		14044	\$0.3777
AWS-EC2-Med-1		\$1,100.55	1	3643	3643	\$0.3021
AWS-EC2-Med-2		\$1,360.55	1	3776	3776	\$0.3603
AWS-EC2-Large-1	Split Data/WAL	\$3,433.57	1	15724	15724	\$0.2184
AWS-EC2-Large-2		\$3,428.57	1	13318	13318	\$0.2574

Results Chart

Smaller Values are Better



Results Chart w/ TPS



Results Analysis

- ▶ RDS
 - ▶ There is a plateau point for a given instance size
 - ▶ Increasing the disk IOPS evaporates money for no performance benefit
 - Unable identify an obvious cause for larger instances: CPU isn't pegged
 - ▶ Would be nice if AWS RDS was smart enough to set limits here for you
 - Small and medium instances better value
- ▶ EC2
 - ► For larger instances (r5.12xlarge) it appears to be trivially easy to beat RDS

Analysis

- For the simple, non-redundant case
 - ▶ Metal is king. Bar none. This laptop matches a m5.xlarge at ~1/5th the cost ratio
 - DBaaS has certain price/performance points where its maybe worthwhile
 - DevOps for this case is easy
 - ▶ Do some analysis
 - ▶ If run-your-own is cheaper (EC2) then do it!
- Future Work: Complex Cases: Multi-master, Replicas
 - ▶ The story is much less clear
 - ▶ How much is DevOps time worth?
 - Know of companies spending substantial engineering \$ to avoid DBaaS
 - ▶ Understand and test the failure cases. Try that with DBaaS.

Summary

- It is OK to solve problems by throwing \$ at them
 - Should at least understand what you are paying for.
 - Are you getting what you need
- The obvious solution may not be the right solution
 - ► EC2 may be better than RDS for large sizes
- Don't boil the ocean
 - Understand your redundancy and DR requirements
 - Build to fit with a little room to grow

Future Work

- Expand the Trade Study
 - Metal Configurations / Co-Lo (rent-a-box)
 - Private Clouds
 - ▶ Nutanix, DC/OS on Metal, Tectonic/OpenShift on Metal
 - Various SDS solutions (OpenEBS, Portworx, Rook, ScaleIO, etc.)
 - ▶ Public Clouds
 - ▶ Azure, Rackspace, Digital Ocean, Google Cloud
- Refine the approach
 - Optimization
 - ▶ Anyone want to help tune PG in docker for each AWS EC2 instance type/disk setup?
 - Additional considerations
 - ▶ Technical
 - ▶ Cost
- Focus is still on TCO for a given set of requirements (\$/month/TPS)

Enhancement Opportunities

- These are some ideas to make PG even better
- DevOps Enhancements
 - Quiesce for filesystems with Atomic CoW snapshots
 - ▶ Matters more for split Data / WAL
 - Some SDS vendors support this (Portworx)
 - ► Something like a CHECKPOINT_AND_PAUSE
 - Consistent Backup to Blob Storage
 - Maybe a pg_cloud_dump
 - ▶ Loadable module
 - Data FEC instead of data checksums
 - Are data checksums really useful if you can't recover

Questions and Discussion

Thanks for your attention.

Questions?

Presentation, Code and Raw Test Data at: https://github.com/jzampieron/pgconf2019/tree/master

Extra Material

Motivation

- Where did this talk come from?
 - Lessons learned running PG in the Cloud
- Balancing DevOps, Data Science and R&D
 - Trying to understand Total Cost of Ownership
 - ▶ NRE
 - ▶ Recurring OpEx vs CapEx
 - ▶ DX
- ▶ How does produce lifecycle fit into these considerations?
- Learning to ask the correct questions