Constraint-Driven Innovation

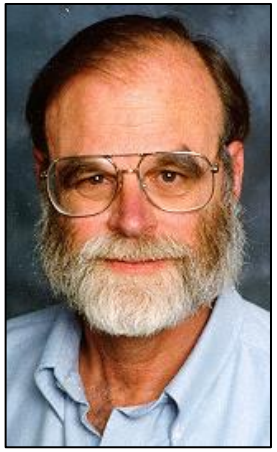
James Hamilton

SVP & Distinguished Engineer

james@amazon.com

January 15, 2024

Why I'm Here



- Jim Gray introduced me to CIDR & HTPS
- CIDR & HPTS have much in common
  - Amazing group of people
  - New ideas not always fully vetted
- My first HPTS was 27 years ago
- The first CIDR was 20 years ago
- Talks & conversations have deeply influenced me
  - Ideas, presentations but, even more, the discussions between sessions

Active Server Availability Feedback

James Hamilton

Microsoft SQL Server
One Microsoft Way
Redmond, WA
USA
JamesRH@microsoft.com

Abstract

The current software development process in common use within industry is inefficient, in that the time required to incorporate results from competitive, beta, and previous releases into new versions available to customers is typically measured in years. Further, the accuracy of customer feedback returned to the development team is frequently weak or incomplete, with samples often drawn from only a small, self-selected set of customers. This paper argues that we can automate this feedback process and, in so doing, drive an order of magnitude improvement in the rate at which software evolves and improves.

development process, which we fully expect will continue. What is less clear is: 1) do systems really need to be this big to meet current customer requirements, and 2) could these systems be evolved more quickly to respond to customer requirements in a more targeted fashion?

Systems growth and development process improvement will continue, but existing processes only allow our current understanding of customer requirements to be translated into software. The improvements do nothing to increase the quality of our understanding of customer requirements nor do they help to tighten the feedback loop between a customer's experience using the product and a subsequent improvement to that product.

An Architecture for Modular Data Centers

James Hamilton
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
+1 (425)703-9972

JamesRH@microsoft.com
http://research.microsoft.com/~jamesrh/

ABSTRACT

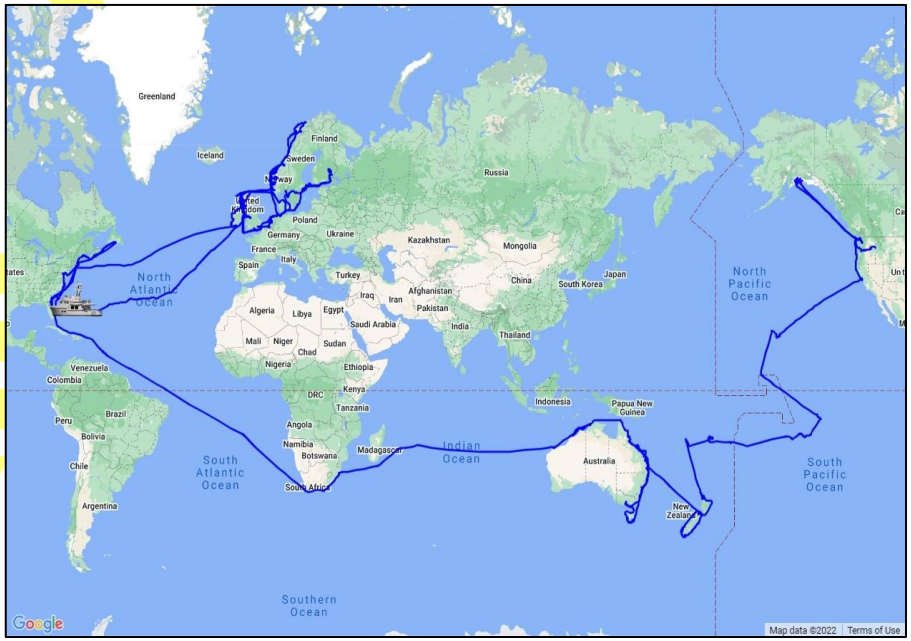
Large numbers of low-cost, low-reliability commodity components are rapidly replacing high-quality, mainframe-class systems in data centers. These commodity clusters are far less expensive than the systems they replace, but they can bring new administrative costs in addition to heat and power-density challenges. This proposal introduces a data center architecture based upon macro-modules of standard shipping containers that optimizes how server systems are acquired, administered, and later recycled.

reliable hardware fails to achieve five 9s, so redundant clusters must be used. Once the software has been written to run efficiently and mask failure over a redundant cluster, then much cheaper and less reliable hardware components can be used as the cluster building blocks without negatively impacting the reliability of the overall service.

Commodity systems substantially reduce the cost of server-side computing. However, they bring new design challenges, some technical and some not. The technical issues include power

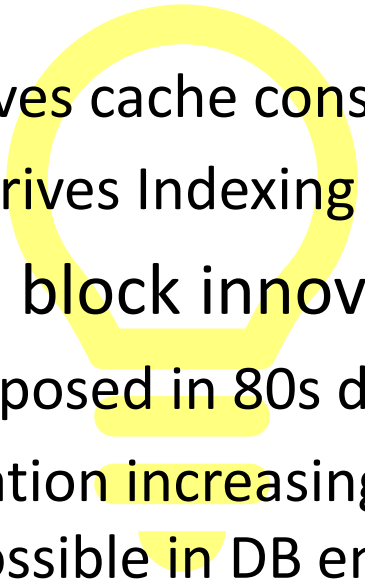
Where Have I Been?

- If CIDR & HPTS are so important, where have I been?
- 2012 to 2021 around the world in a small boat
  - Worked full time at AWS
  - Only in Seattle 3 to 4 times/year
  - Incredible experience
  - Memorable satellite bill :-)
- Great to be back at CIDR!



Constraint-Driven Innovation

- Constraints force innovation
  - Incredibly poor HDD IOPS has driven storage engine & logging innovation
  - Memory Chasm drives cache conscious data structures
  - Poor storage B/W drives Indexing & materialized views
- But constraints also block innovation
  - In-memory DBs proposed in 80s didn't happen for 20 years
  - H/W pace of innovation increasing & fundamentally changing what is possible in DB engines
- Talk focuses on innovations possible as constraints fall
  - I see great opportunity in H/W DB acceleration



aws

Open Source Databases

- Constraint: Research community access to DB source
- Open source available, but years to gain critical mass
  - 1995: MySQL
  - 1996: PostgreSQL
- Many smart people innovating on these code bases
  - Originally not competitive with commercial DBs
  - Price/performance attractive & technology improved rapidly
- Constraint lifted: Healthy open source ecosystem
  - Enables broad R&D contributions outside commercial DBs

2005: One Size Doesn't Fit All

- Stonebraker thesis: App-specific DBs 10x faster
  - Simple ideas can deliver the most profound impact
  - Unleashed 2 decades of innovation
- Constraint: Complex admin limits most to 1 DB variant

"One Size Fits All"
An Idea Whose Time Has Come and Gone

by

Michael Stonebraker

"One Size Fits All": An Idea Whose Time Has Come and Gone

Michael Stonebraker
Computer Science and Artificial Intelligence Laboratory, M.I.T., and StreamBase Systems, Inc.
stonebraker@csail.mit.edu

Uğur Çetintemel
Department of Computer Science Brown University, and StreamBase Systems, Inc.
ugur@cs.brown.edu

Application-Specific DBs

- Constraint: Admin complexity of multiple DBs
- Cloud computing reduces admin overhead
  - Cloud service handles DB administration
  - Less overhead to use workload-optimized DBs
- Innovation: >19 database services at AWS
  - OS Relational: Aurora, MySQL, PostgreSQL, MariaDB
  - Commercial: SQL Server, Oracle, DB2
  - DW/Analytics: Redshift, Athena
  - NoSQL: DyamoDB, DocumentDB, Keyspaces
  - Graph DB: Neptune, Tinkerpop,
  - Time Series: TimeStream
  - In-Memory: ElastiCache, OpenSearch, MemoryDB, Redis

2011: Systems & DB Evolve Together

- Constraint: DB scaling
- MapReduce brought new DB/Analytics researchers
  - MapReduce scales but brute force
  - No SQL & missing 30 year-old optimizations
  - Testy relationship at first
- Constraint lifted:
  - MapReduce/Spark & RDBMS evolve towards each other
  - Increased pace of innovation

MapReduce: A major step backwards | The Database Column

on Jan 17 in Database architecture, Database history, Database innovation posted by DeWitt

[Note: Although the system attributes this post to a single author, it was written by David J. DeWitt and Michael Stonebraker]

On January 8, a Database Column reader asked for our views on new distributed database research efforts, and we'll begin here with our views on MapReduce. This is a good time to discuss it, since the recent trade press has been filled with news of the revolution of so-called "cloud computing." This paradigm entails harnessing large numbers of (low-end) processors working in parallel to solve a computing problem. In effect, this suggests constructing a data center by lining up a large number of "jelly beans" rather than utilizing a much smaller number of high-end servers.

For example, IBM and Google have announced plans to make a 1,000 processor cluster available to a few select universities to teach students how to program such clusters using a software tool called MapReduce [1]. Berkeley has gone so far as to plan on teaching their freshman how to program using the MapReduce framework.

As both educators and researchers, we are amazed at the hype that the MapReduce proponents have spread about how it represents a paradigm shift in the development of scalable, data-intensive applications. MapReduce may be a good idea for writing certain types of general-purpose computations, but to the database community, it is:

1. A giant step backward in the programming paradigm for large-scale data intensive applications
2. A sub-optimal implementation, in that it uses brute force instead of indexing
3. Not novel at all — it represents a specific implementation of well known techniques developed nearly 25 years ago
4. Missing most of the features that are routinely included in current DBMS
5. Incompatible with all of the tools DBMS users have come to depend on

Perspectives

Home Boat blog About Perspectives

MapReduce: A Minor Step Forward

Dave Dewitt and Michael Stonebraker posted an article worth reading yesterday titled: MapReduce: A Major Step Backwards (Thanks to Kevin Merrit and Sriram Krishnan for sending this one my way). Their general argument is that MapReduce isn't better than current generation RDBMS which is certainly true in many dimensions and it isn't a new invention which is also true. I'm not in agreement with the conclusion that MapReduce is a major step backwards but I'm fully in agreement with many of the points building towards that conclusion. Let's look at some of the major points made by the article:

1. MapReduce is a step backwards in database access

In this section, the authors argue that schema is good, separation of schema and application are good, and high level language access is good. On the first two points, I agree schema is good and there is no question that application/schema separation has long ago proven to be a good thing. The thing to keep in mind is that MapReduce is only an execution framework. The data store is GFS or sometimes Bigtable in the case of Google or HDFS or HBase in the case of Hadoop. MapReduce is only the execution framework so it's not 100% correct to argue that MapReduce doesn't support schema – that's a store issue and it is true that most stores that MapReduce is run over don't implement these features today. I argue that a separation of execution framework from store and indexing technology is a good thing in that MapReduce can be run over many stores. You can use MapReduce over either BigTable (which happens to be implemented on GFS) or

Type here

James Hamilton

aws

Separation of Compute & Storage

- Constraint: Compute & storage on same server
  - Most on-premise DBs are monolithic
  - QP, Optimizer, Execution engine, storage engine, & storage all on same server
- Cloud enables separation
  - More servers yields little increased complexity to customers
- Innovations enabled:
  - Query compilation & execution can scale & fail independently of storage
  - Storage can span availability zones (datacenters) for much higher durability and availability

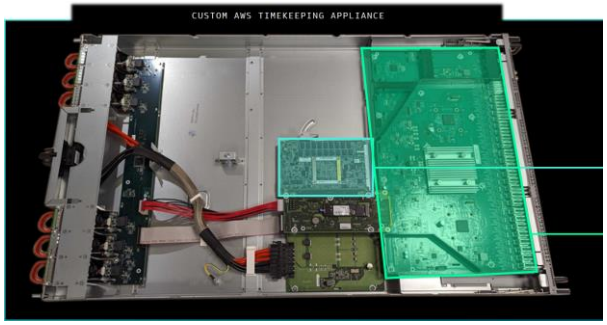
Object Store as DB Primary Store

- Constraint: Shared disk architectures difficult to scale
- Cloud object store scales wildly beyond any shared disk system
  - Latency constraint remains: old DB tricks to hide HDD latency also effective at hiding object store latency
- Innovations enabled:
  - Rapid new cluster creation, resize, ~100msec fault recovery
  - Read replicas can be created at will
  - Node failure recovery easy and fast
  - Multiple different-sized clusters can operate on same data
- 2012: Snowflake delivered cloud-optimized database
  - Storage layer was Amazon S3
  - Highly durable storage (11 9s design)

aws

High-Precision Clocks

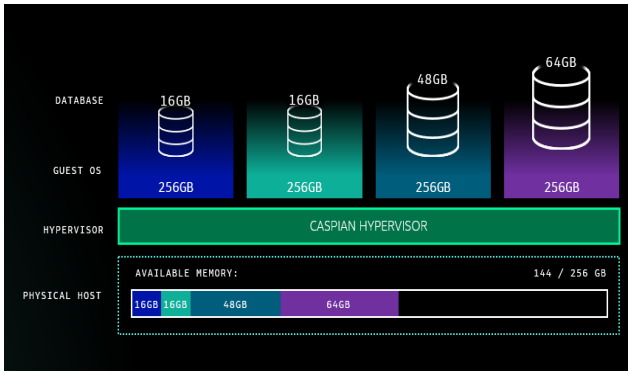
- Constraint: Cross-server clock skew
- Precision clocks now economic
  - Google TrueTime <7 ms for Spanner
  - Amazon Time Sync Service: <100 usec
  - Both good enough & both easy to improve



- Bounded clock error doesn't roll back CAP Theorem
  - But enables numerous very useful innovations
  - Partitions still (rarely) occur in private networks
- Innovations enabled:
  - Consistent cross datacenter/continent MVCC snapshots
  - Low clock skew allows practical & useful transaction rates
- Much higher clock accuracy easily within reach supporting much lower latency transactions

Cooperative Memory Oversubscription

- Constraint: Memory doesn't virtualize well
  - The industry has successfully virtualized storage
  - And distributed workloads over multiple servers
  - But memory virtualization hasn't been effective
    - NUMA near-to-far memory ratios >2.0 super hard to hide
- DB out-of-memory impact high
  - So, memory usually overprovisioned
- Innovation: Oversubscribe memory
  - DB can request more/less memory
  - Manage resource conflict via DB & O/S live migration
  - Innovation used in Amazon Aurora DB service

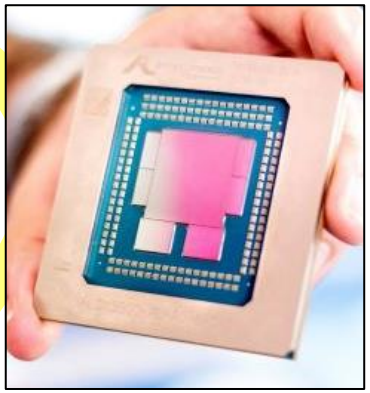


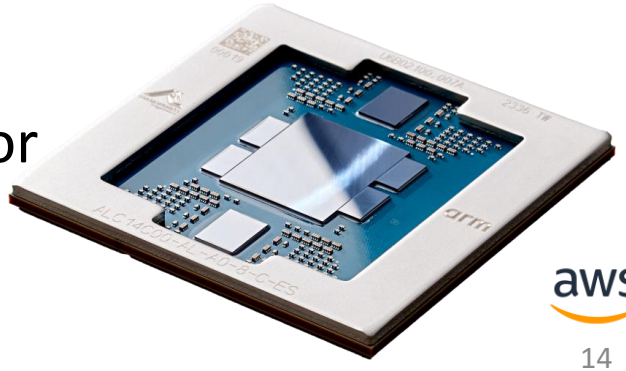
Database Cost

- Constraint: Database HW & SW costs
- Traditional transactional systems scale with business growth
  - Purchases, ad impressions, pages served, etc.
  - Machine-to-machine transactions scale limited only by value of transaction & cost to process
- Warehouse & analytical use scales inversely with cost
  - Lower costs supports more data & deeper analysis
- Constraint lifted: Rapidly declining cost of computing
  - H/W innovation & cloud computing economies of scale

Cloud Scale Feeds Innovation

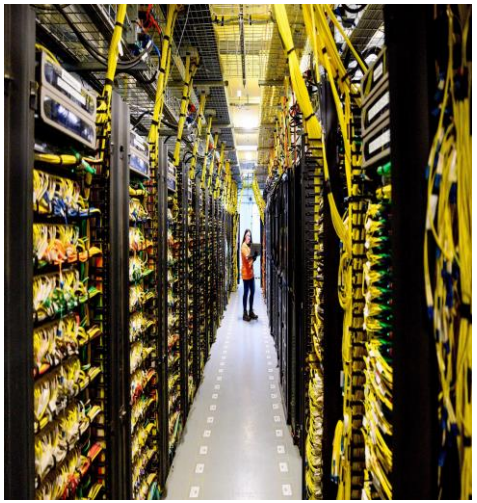
- Cloud scale supports H/W R&D, which drives innovation, which further drives scale
- AWS examples:
  - Custom server designs
  - Custom network designs
  - Custom semiconductors
    - Nitro service, storage, security & network offload
    - Graviton server CPU
    - Inferentia ML inference processor
    - Trainium ML training processor





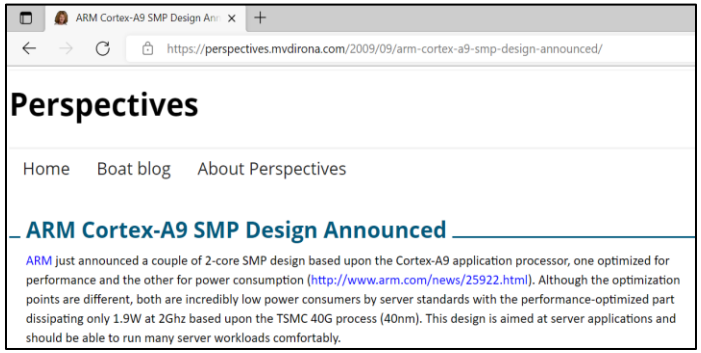
Custom Servers

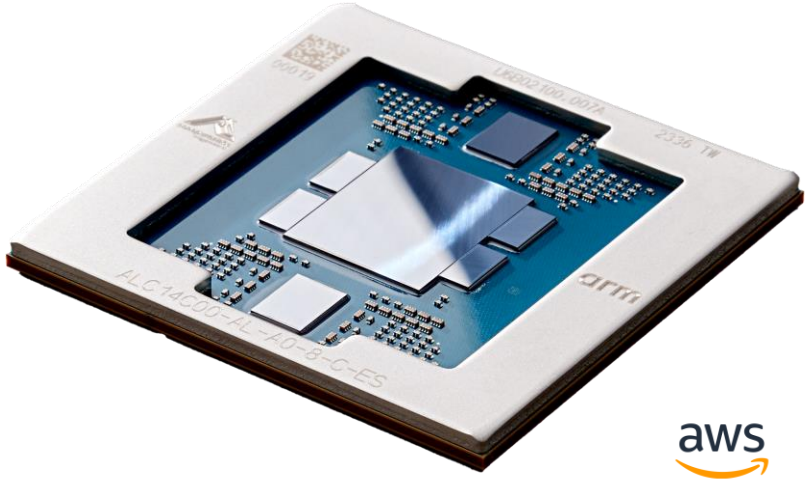
- AWS has designed & developed custom servers for more than a decade
  - Reduced cost
  - Multi-source contract manufactures
  - Full control of supply chain
  - Proprietary security features
  - Key: Workload-specific optimization
    - Volume drives specialization
    - ML is most radical example & shows what is possible for DB



Custom General Purpose CPUs

- *"AWS Custom H/W"* doc review in 2013
  1. Arm will yield a great server processor
  2. Server innovation is moving on-package
- Device & IoT volume supports R&D
  - In 2021 Arm crossed 215B processors
  - I first blogged in 2009
- Graviton 4: +30% perf over Graviton 3
  - 74B transistors
  - 96 Arm Neoverse V2 Cores
  - 64k L1 / 2MB L2
  - 12 DDR5 lanes
  - 7-die multi-chip package



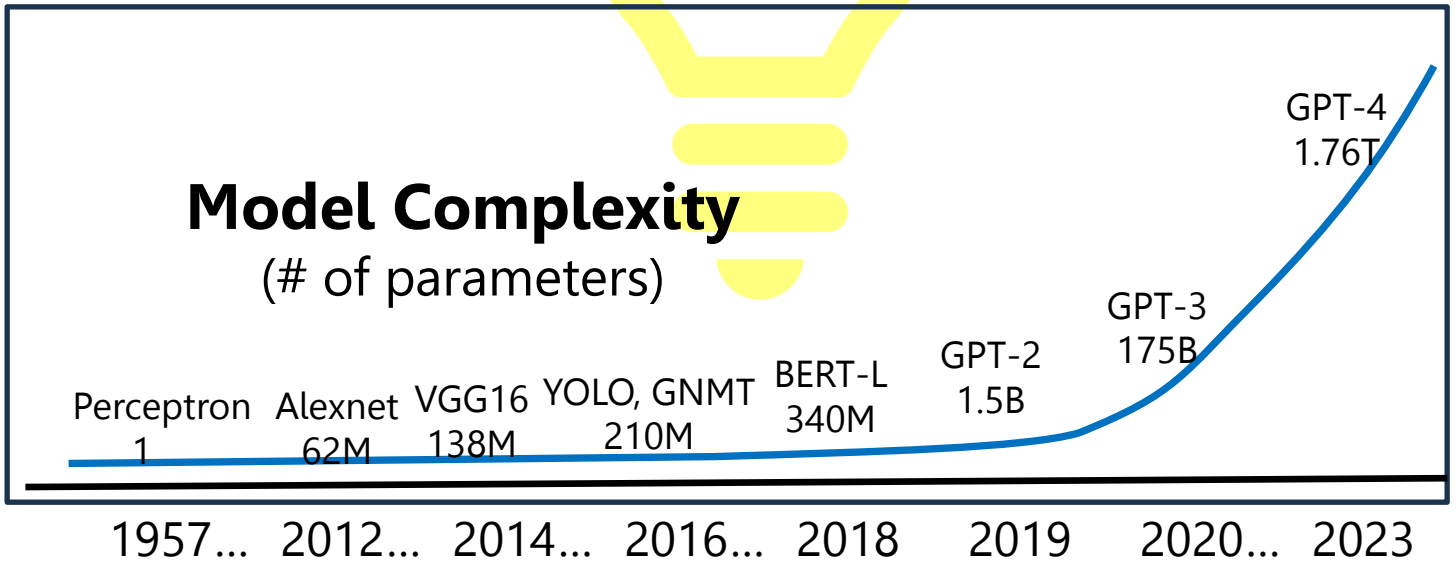


Custom ASIC: Nitro

- First AWS ASIC
  - Private server in every server
- Nitro features:
  - Network H/W offload with RDMA
  - Storage H/W acceleration
  - H/W protection & security
  - Hypervisor offload
- Many parallels with mainframe design points
  - I/O offload to dedicated Channel Processors
  - RAS & admin offload to Service Processor
- Over 20 million installed

Machine Learning Training

- Technology growth of >2x brings mass innovation
  - Most I've seen since data-warehousing in 90s
- Important to database in 3 ways:
  - ML used to implement DB features
  - ML features most efficient when integrated into DB
  - Shows what is possible with domain-specialized hardware

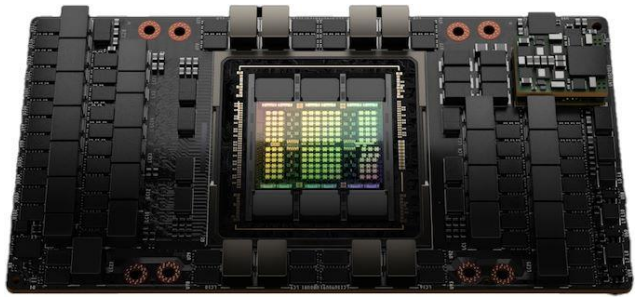


Amazon Q

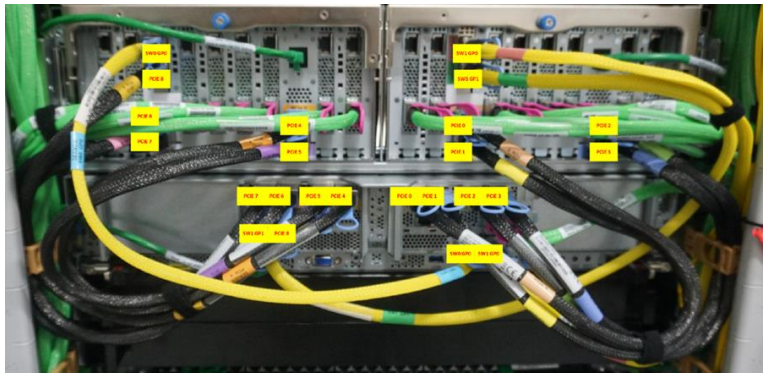
- AWS GenerativeAI Assistant
  - In the console, IDE, and documentation
  - Trained on 17 years of AWS knowledge
- Explore new AWS capabilities, technologies, & architect solutions
  - Code Whisperer to assist code development
- Java version upgrades
- Windows .Net to Linux migrations
- Reporting & Analytics: Q for Quicksite
- Find & summarize docs across org
- Troubleshoot, build new features, and upgrade apps

Current Gen Nvidia: H100

- EC2 P5 Instance type
  - 6RU, multi-chassis server
  - Peak power 12.6 kW
- 8 PCIe attached NVIDIA H100 GPUs
  - Massive 814 mm^2 in TSMC N4 Process
  - 18,432 Cuda cores each – 147,456 cores across full server
  - 3,026 TFLOPS @ FP8
  - ~$30,000 each H100
  - 1/3 of million dollar servers
- Mem: 640GB HBM3 + 2,000GB
- 3,200 Gbps RDMA network
- 8x 3.84 TB NVMe SSDs



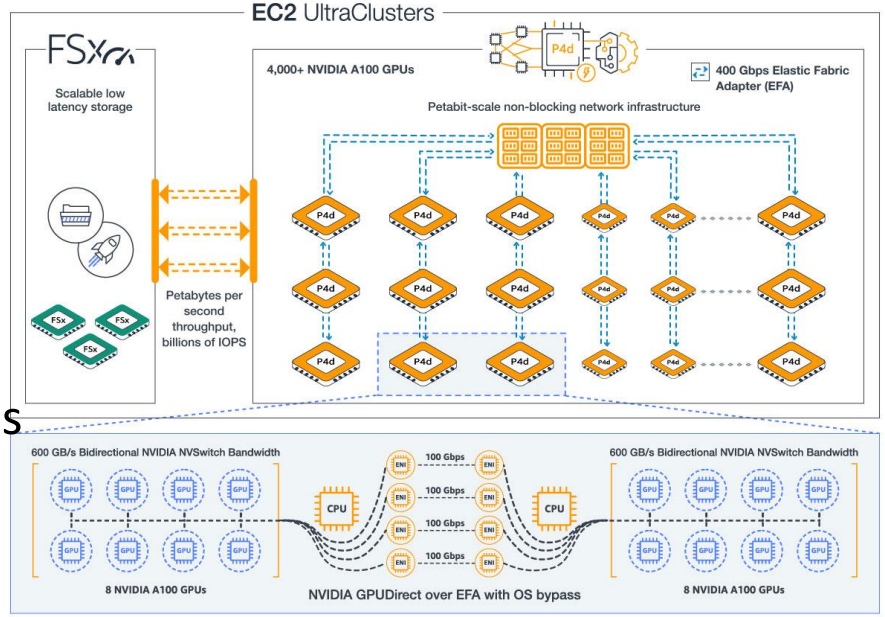
Nvidia H100



EC2 P5 Instance

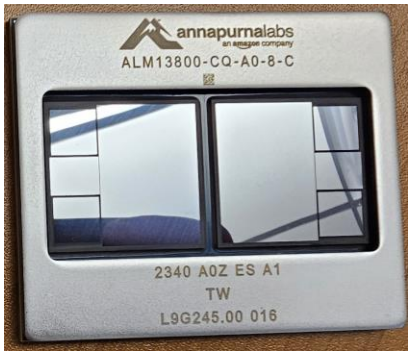
High Scale ML Training

- Example training run from earlier this year (1 gen old)
- 200B param dense model trained on 4T tokens
  - 1,720 P4d nodes (13,760 Nvidia A100)
- 48 day training time (including node faults)
  - ~$65M training cost
  - 10.3MW & 11.9 GWhr
- Strategy:
  - Tensor || within 8 GPU server
  - Pipeline ||with depth of 40 nodes
  - Data || across 43 pipelines
  - Global batch size ~4M tokens
- Training runs soon to cross $1B



Trainium Servers

- EC2 Trn1 instance type
  - Announced 2020
  - 16 Trainium ASICs
  - 512GB HBM2 memory
  - 800Gbps networking
  - Stochastic rounding
- EC2 Trn2 instance type
  - Announced 2023
  - 16 Trainium2 ASICs
  - 1,536GB HBM3 memory
  - 3,200Gbps networking
  - Stochastic rounding
- Similar application specialization could apply to DB



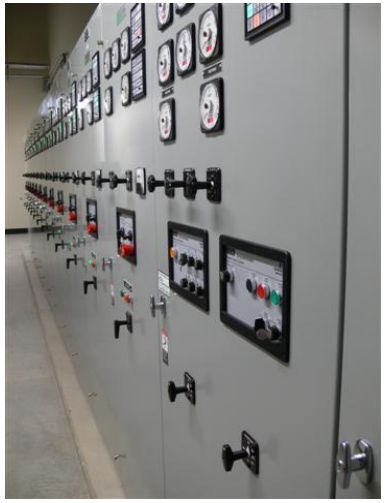
Storage & Memory B/W lagging CPU

	CPU	DRAM	LAN	Disk
Annual bandwidth improvement (all milestones)	1.5	1.27	1.39	1.28
Annual latency Improvement (all milestones)	1.17	1.07	1.12	1.11

- Constraint: Power consumption & data availability
- CPU bandwidth out-pacing memory & storage
  - Disk & memory getting “further away” from CPU
  - Powered CPU cores have no value without data
  - All workloads are “data constrained”
- These constraints aren’t going away

Source: Dave Patterson: Why Latency Lags Bandwidth and What It Means to Computing

Limits to Computation



- Processor cycles are cheap & getting cheaper
- What limits the application of infinite cores?
  1. **Data**: inability to get data to processor fast enough
  2. **Power**: cost rising and will dominate
- Most sub-Moore attributes need most innovation
  - Infinite processors require infinite power
  - Getting data to processors in time to use next cycle:
    - Caches, multi-threading, ILP,…
    - All techniques consume power
    - All off chip techniques consumes considerably more power
- Power & data movement key constraints to DB
  - We've enjoyed a wonderful period of constraint removal
  - **But we'll need to innovate around these unchanging constraints**

Closing

- This is the golden age of DB innovation
- Many traditional constraints have fallen
  - Cloud computing
  - Open source
  - Hardware specialization
- ML central to DB going forward
  - Optimizers, index creation, materialized views, …
  - ML customer-facing features
- ML shows opportunities with H/W specialization
  - Big database innovations still coming

Constraint-Driven Innovation

Slides: mvdirona.com/jrh/work
Email: james@amazon.com
Blog: perspectives.mvdirona.com