TidalScale

Scale | Simplify | Optimize | Evolve

- Increasing Flash Throughput for Big Data Applications (Data Management Track)

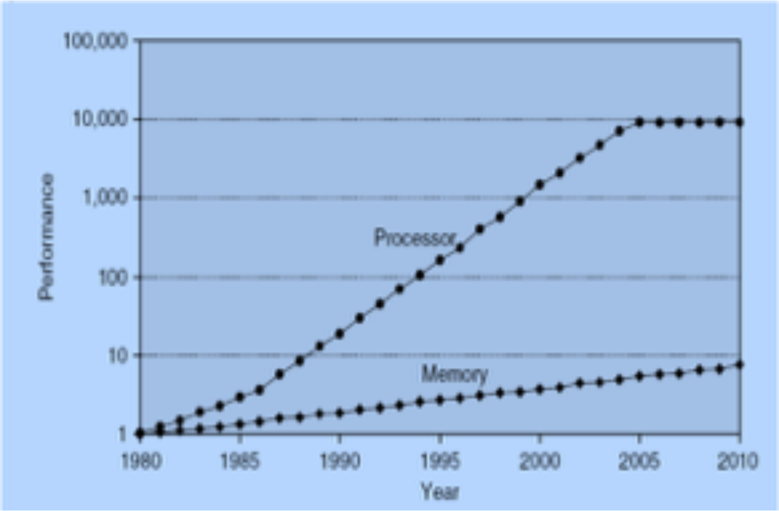
- **Industry Context**
- Addressing the challenge
- A proposed solution
- Review of the Benefits

Industry Context

- Data sizes for data under management are monotonically increasing
  - Who wants less data?
- Our appetite for analysis is monotonically increasing
  - Do you **think,** or do you **know?**
  - Trend toward evidence-based management
- Our appetite for speed is monotonically increasing
  - Who wants questions answered more slowly?
  - Hence the industry interest in in-memory data management systems
- **Our overall ability to manage complexity is <u>not</u> increasing**

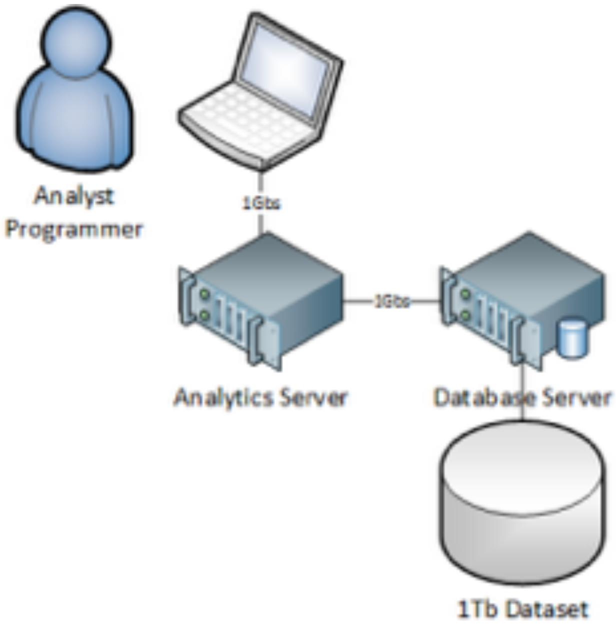
Processor / Memory Context

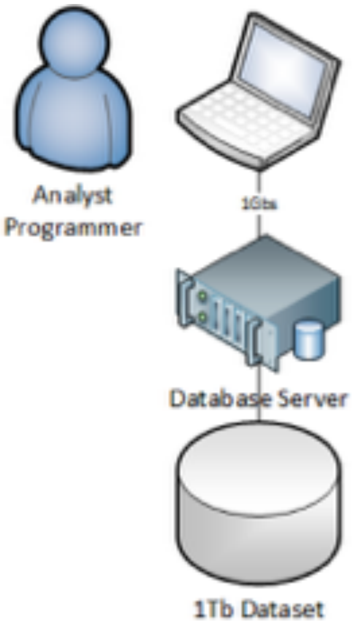
- Processor speeds are limited
- Processor core density has been increasing at a healthy rate
- Memory density is increasing (but at a lower rate than core density)!
- Therefore, the memory/core ratio is going in the wrong direction!
- We haven't significantly changed the memory/storage hierarchies for decades
    - Interconnects are getting faster – as fast as memory access?
    - memory access is slow
    - caches are fast!



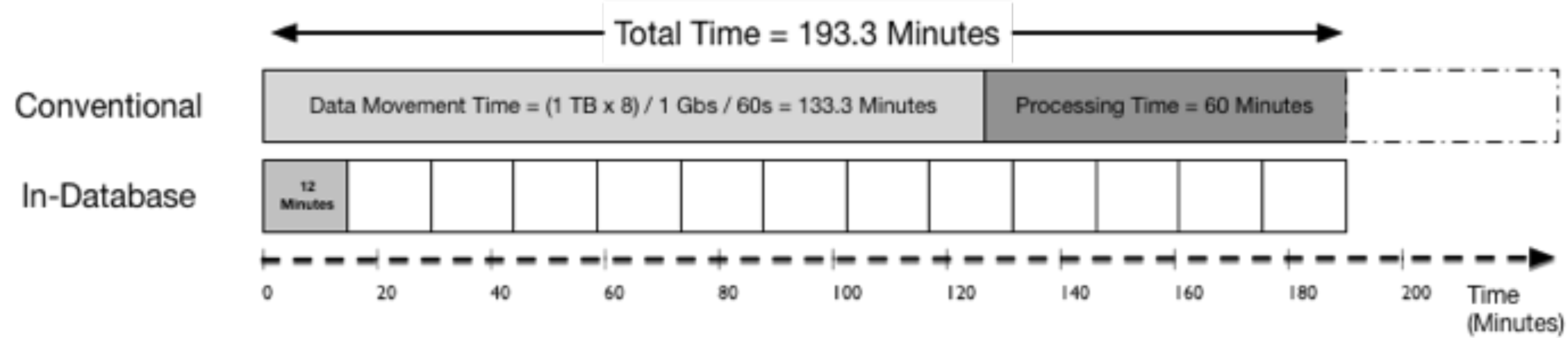
Hennessy, John L.; Patterson, David A. (2011-10-07). Computer Architecture: A Quantitative Approach (The

Which is faster, and why?





Which is faster, and why?



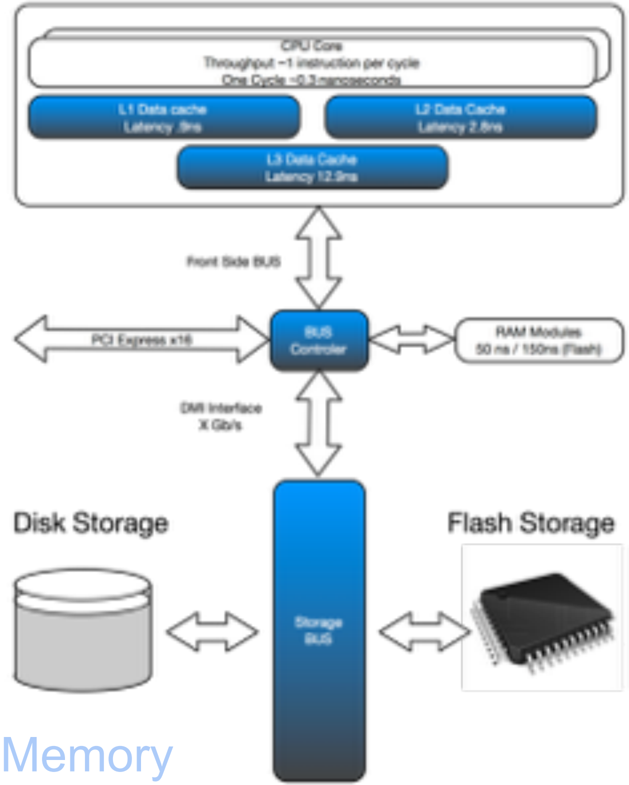
Moving data is the most time-consuming activity, reducing data movement is key. I/O is the enemy

The Memory Hierarchy in Human Terms

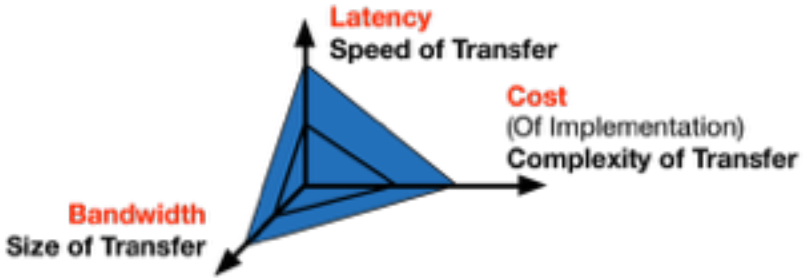
(.3ns = 1s)

Event	Latency	Scaled
1 CPU Cycle	0.3 ns	1 s
Level 1 Cache Access	0.9 ns	3 s
Level 2 Cache Access	2.8 ns	9 s
Level 3 Cache Access	12.9 ns	43 s
Main Memory Access (DRAM, from CPU)	50.0 ns	3 min
Memory over Ethernet	3.2 µs	3.2 hours
CPU Context State Transfer	6.0 µs	6.0 hours
Flash SSD (PCI-e)	4.7 ms	5 months
Rotational disk I/O	1-10 ms	1-12 months
Internet: San Francisco to New York	40 ms	4 years
Internet: San Francisco to United Kingdom	81 ms	8 years
Internet: San Francisco to Australia	183 ms	19 years
TCP packet retransmit	1-3 s	105-317 years

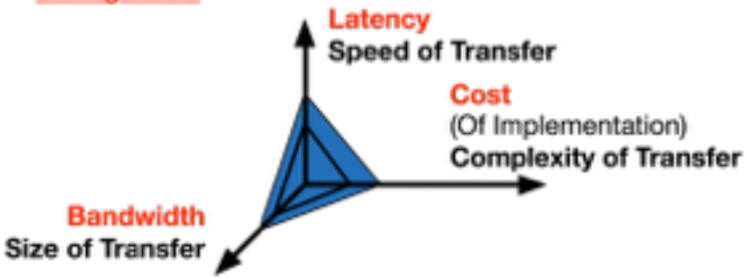
I/O Bus Choice
Memory Verses Storage



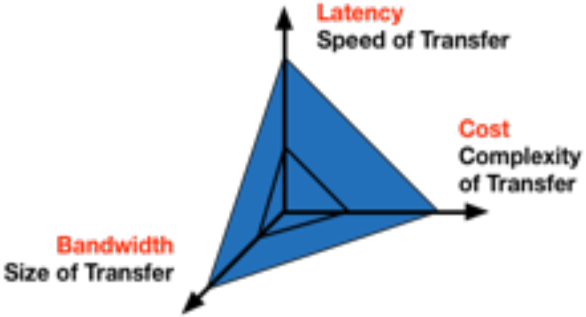
Memory BUS

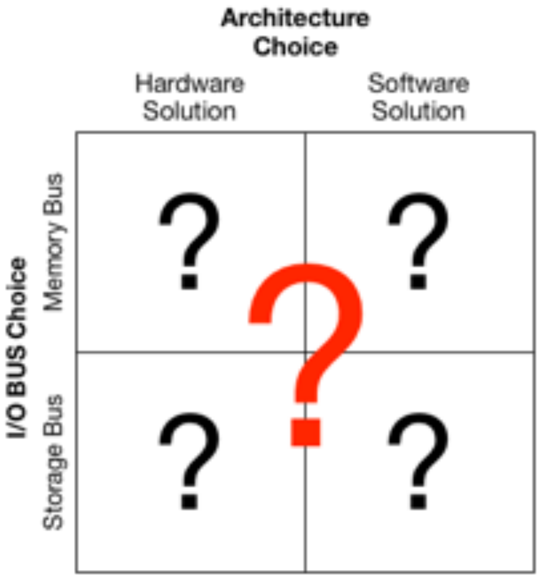


Storage BUS



Choices and Tradeoffs

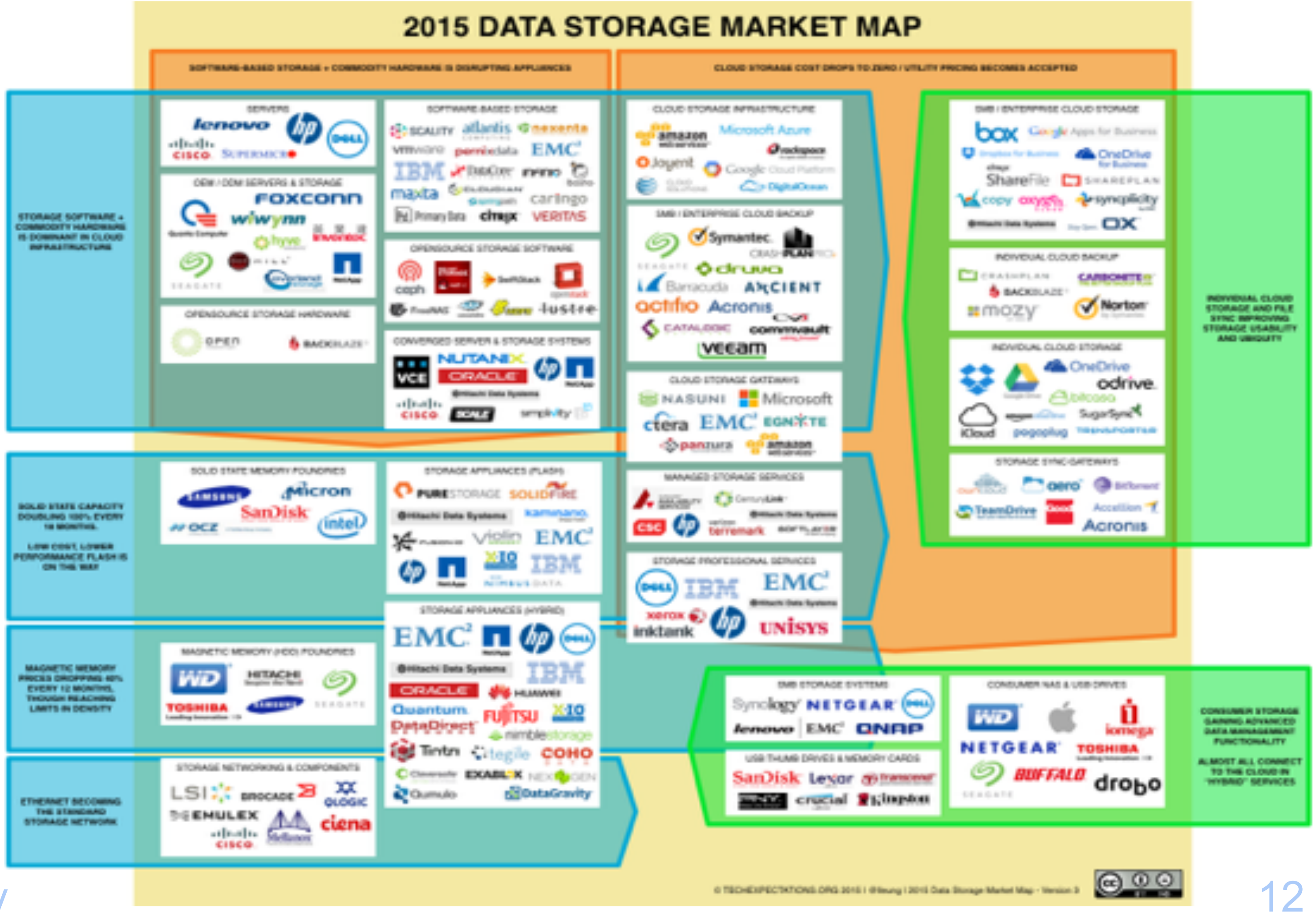




- Industry Context
- **Addressing the challenge**
- A proposed solution
- Review of the Benefits

The Challenge

- How big is “Big”, and what kind of computer architecture do you need to optimally solve a Big Data problem?
- Big Data applications may have irregular and unpredictable data access patterns
  - Efficient partitioning of data and queries can be challenging
  - Data can change, and structurally vary characteristics over time
- Big Data applications often must solve system problems
  - Coordination of computing resources (Storage, Partitions, Networks, CPU’s)
  - People who have deep analytic skills are often required to design system resource strategies, perhaps not the greatest use of their time and expertise


2015 DATA STORAGE MARKET MAP

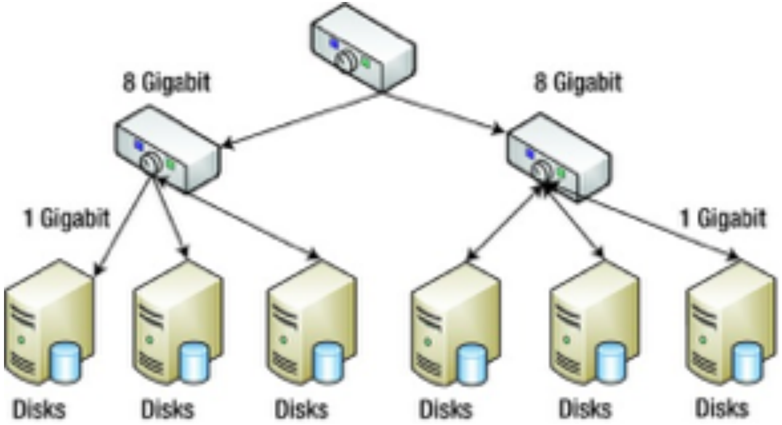
The Three V's of Big Data

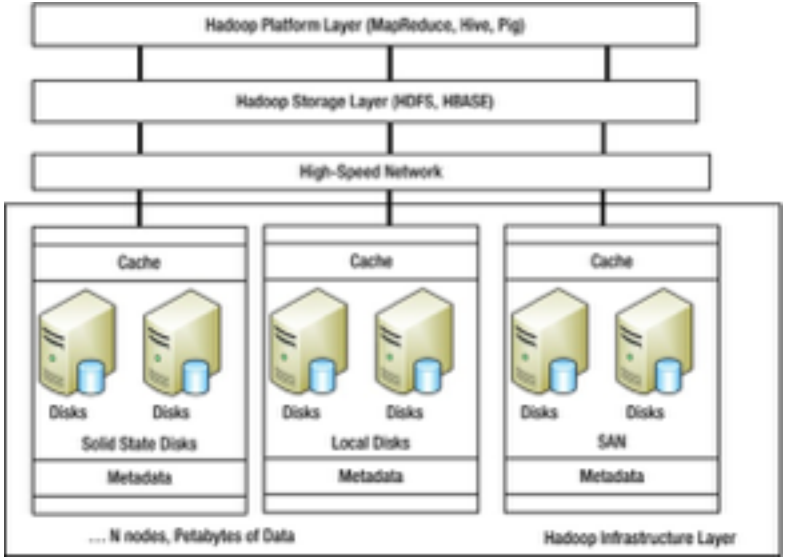
Src: https://medium.com/deepend-indepth/know-your-audience-better-than-asio-4802839c3fd3

Data Locality is the real problem not, Volume, Velocity or Variety

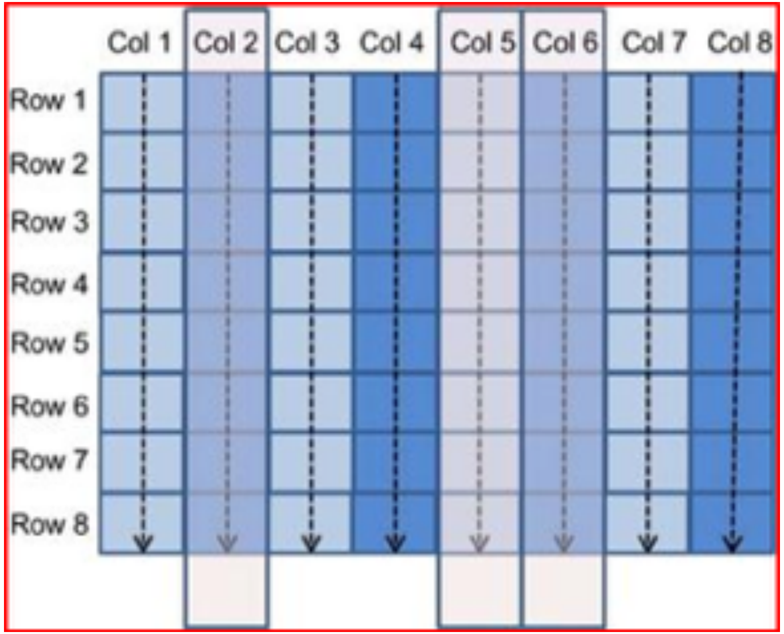
- The Impact of Data Locality
  - **Volume:** The further the data has to move the worse your performance will be.
  - **Velocity:** The more complex the transformations the worse your performance will be.
  - **Variety:** The more data components you look at the worse your performance will be.

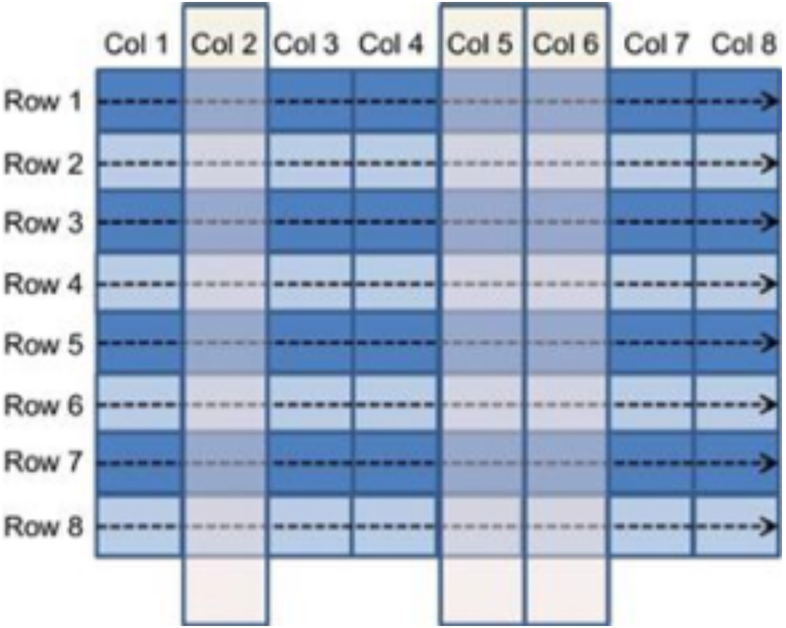
An Example: Where to optimize?





A Simple Improvement: Column verses Row data architecture





The End Goal

- **In-Memory Performance**
  - The world wants data to be in-memory, but hasn't been able to get it.
  - Historically the industry has scaled applications to fit on available computer hardware.
  - The industry needs to move to scale the hardware to fit the application.
- **Linear System Scalability**
  - We need systems that enable hardware to scale organically using low cost commodity servers at linear cost, as customer needs evolve.
  - We need to enable applications to achieve superior in-memory performance using inexpensive unmodified hardware.
- **Reduced Software Development Costs**
  - We need to use off-the-shelf, unmodified Linux.
  - We require no changes to applications or database software.
  - We require systems that optimize automatically, and learn over time.

- Industry Context
- Addressing the challenge
- A proposed solution
- Review of the Benefits

Choice: scale up or scale out?

- Both scale up and scale out can be expensive
- When all you have is a hammer…
  - Every problem looks like a nail
- Today we rely almost exclusively on "scale out" systems
  - Because that's the main way we add processors and memory
  - ➔ Shard the data, intelligently target the queries – time consuming
  - It's not easy to query partitioned databases
    - What is the best way to do it?
    - Moving data is time-consuming
    - And you might have to change it

- **What if you could build systems that "Scale up and Out"?**

Opportunity:

- Enable application developers to focus time on **applications,** not the **systems** required to run them

- Move the coordination and management tasks to the “Operating System” (broadly based)

- Remove the requirement that managing resources be part of the Big Data application

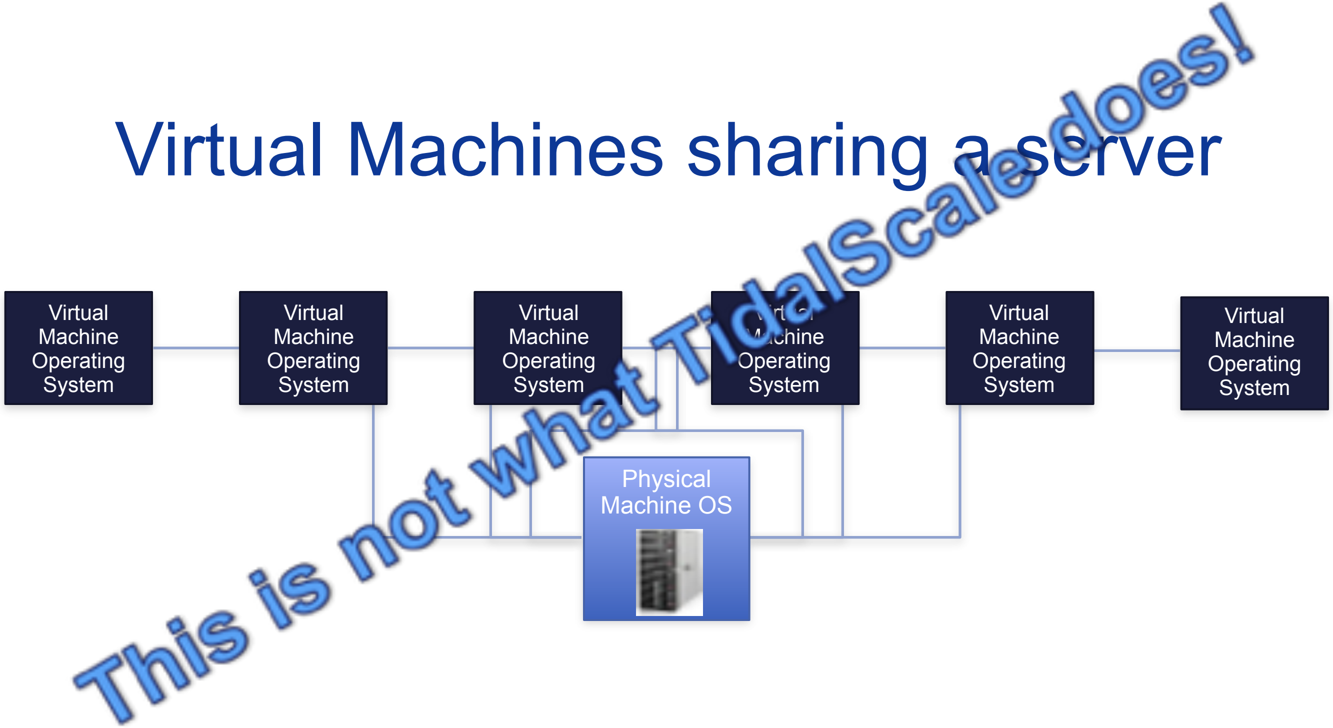
How?

- 45 years ago we figured out how to virtualize memory using the locality principle*
- Questions:
  - Could locality be applied ubiquitously across our computing infrastructure?
  - **How might we apply locality to all compute resource types automatically & dynamically?**

* P.J. Denning

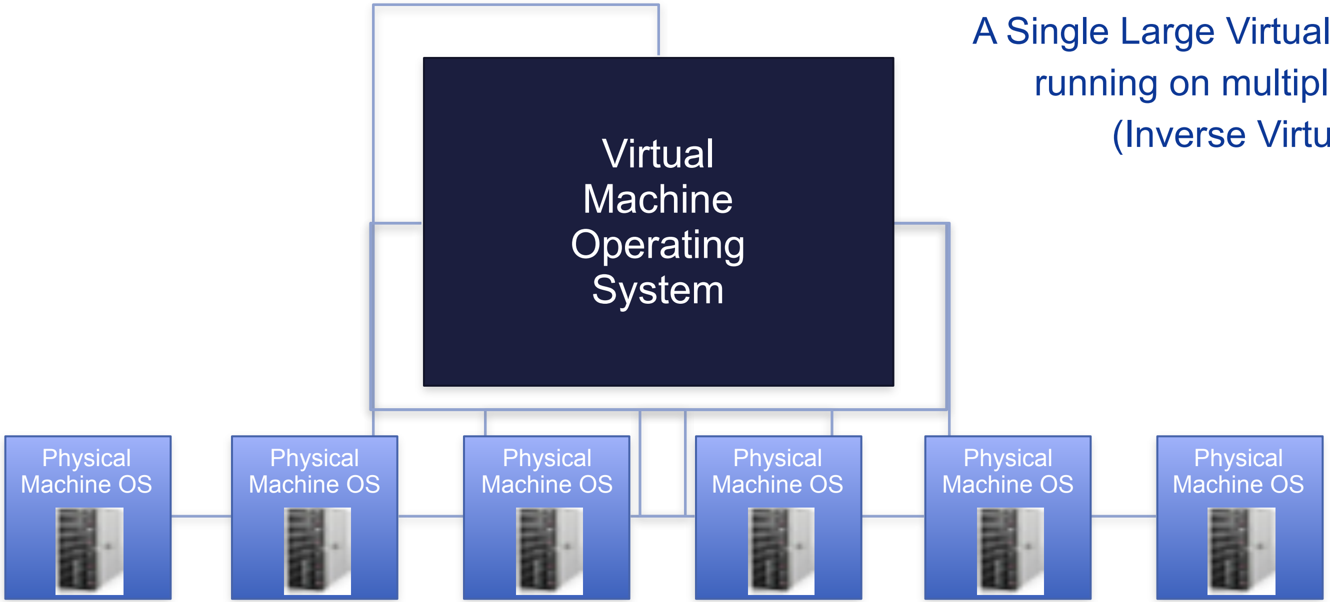
Traditional view of virtualization

Virtual Machines sharing a server

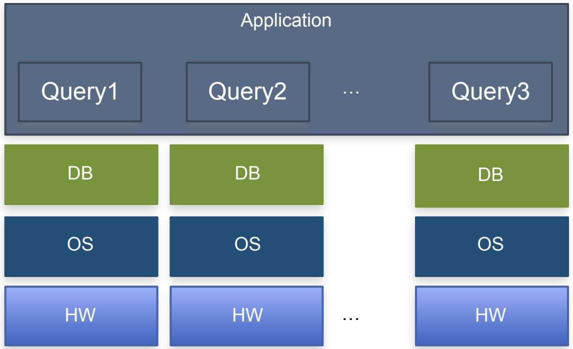


TidalScale
Provides hardware aggregation

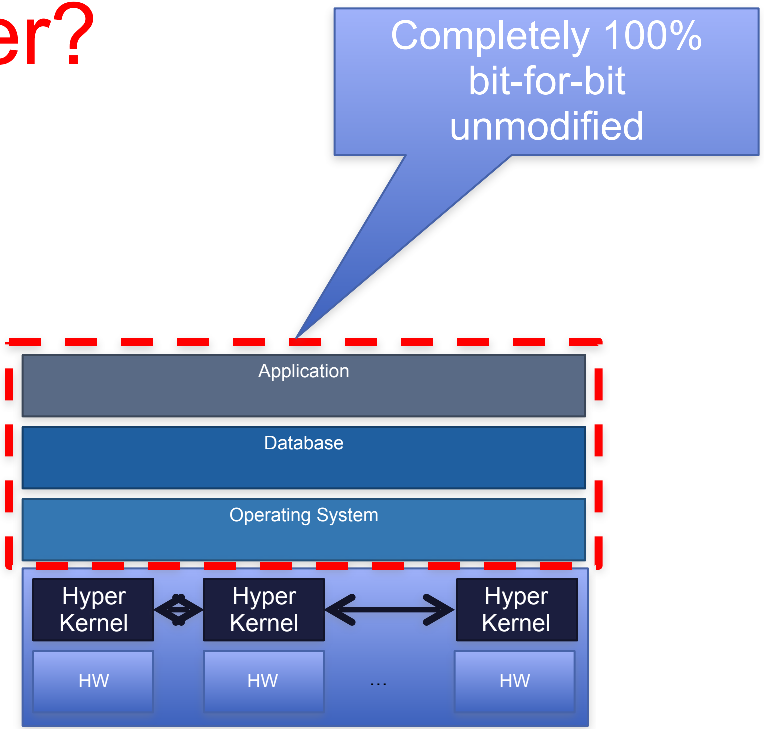
A Single Large Virtual Machine
running on multiple servers
(Inverse Virtualization)



How is this better?



Today's Model



TidalScale Model

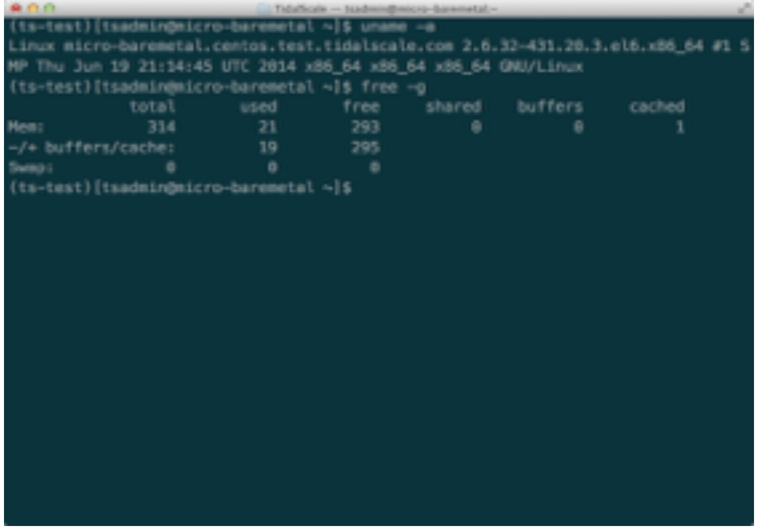
What enables the solution?
The Memory Hierarchy in Human Terms

	Event	Latency	Scaled
TidalScale Operating Zone	1 CPU Cycle	0.3 ns	1 s
	Level 1 Cache Access	0.9 ns	3 s
	Level 2 Cache Access	2.8 ns	9 s
	Level 3 Cache Access	12.9 ns	43 s
	Main Memory Access (DRAM, from CPU)	50.0 ns	3 min
	Memory over Ethernet	3.2 µs	3.2 hours
	CPU Context State Transfer	6.0 µs	6.0 hours
Memory Cliff	Flash SSD (PCI-e)	4.7 ms	5 months
	Rotational disk I/O	1-10 ms	1-12 months
	Internet: San Francisco to New York	40 ms	4 years
	Internet: San Francisco to United Kingdom	81 ms	8 years
	Internet: San Francisco to Australia	183 ms	19 years
	TCP packet retransmit	1-3 s	105-317 years

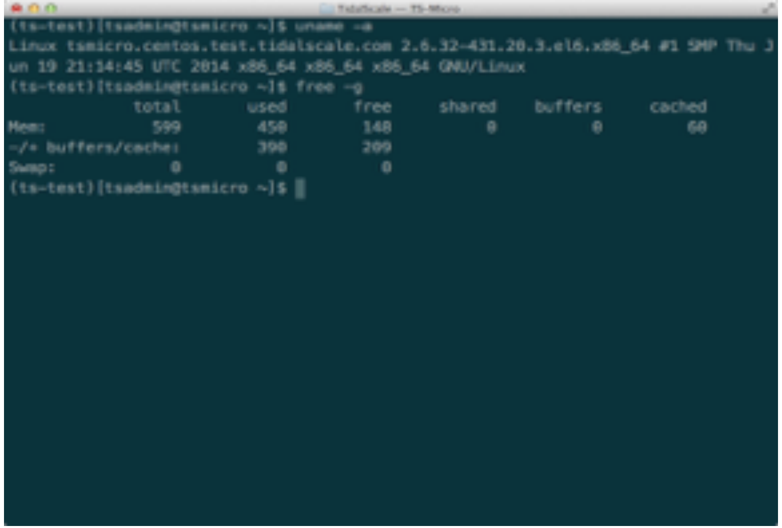
(.3ns = 1s)

Why?
Creates a unique scalable solution experience:

User experience bare metal



User experience TidalScale



Real Screen Shots

Hardware Example

System features

Admin node	1
Worker nodes	25
Total Memory	3.2TB
Total Cores:	150
Network	1/10GbE
Storage	FreeNAS, xTB

Components

1x Admin node (A003)

Colfax CX1260i-X6 Haswell, E5-2603V3 6C, 16GB

25x Worker nodes

Colfax CX1260i-X6 Haswell, E5-2603V3 6C, 128GB

1x 1G switch

2x 10G switch (S009, S010) Mellanox

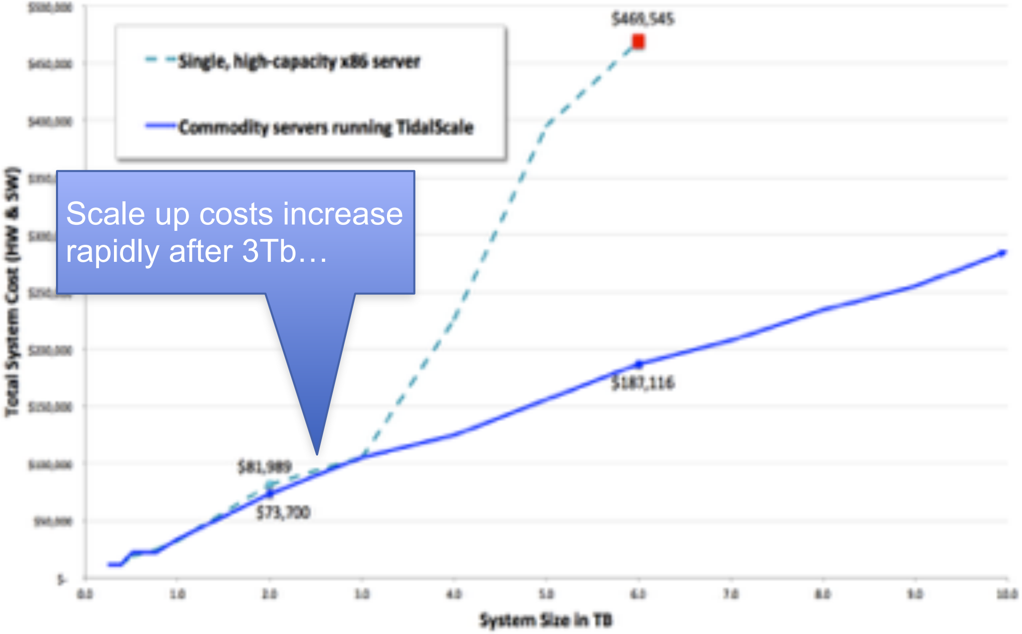
1x NAS

TidalScale - No Changes Required

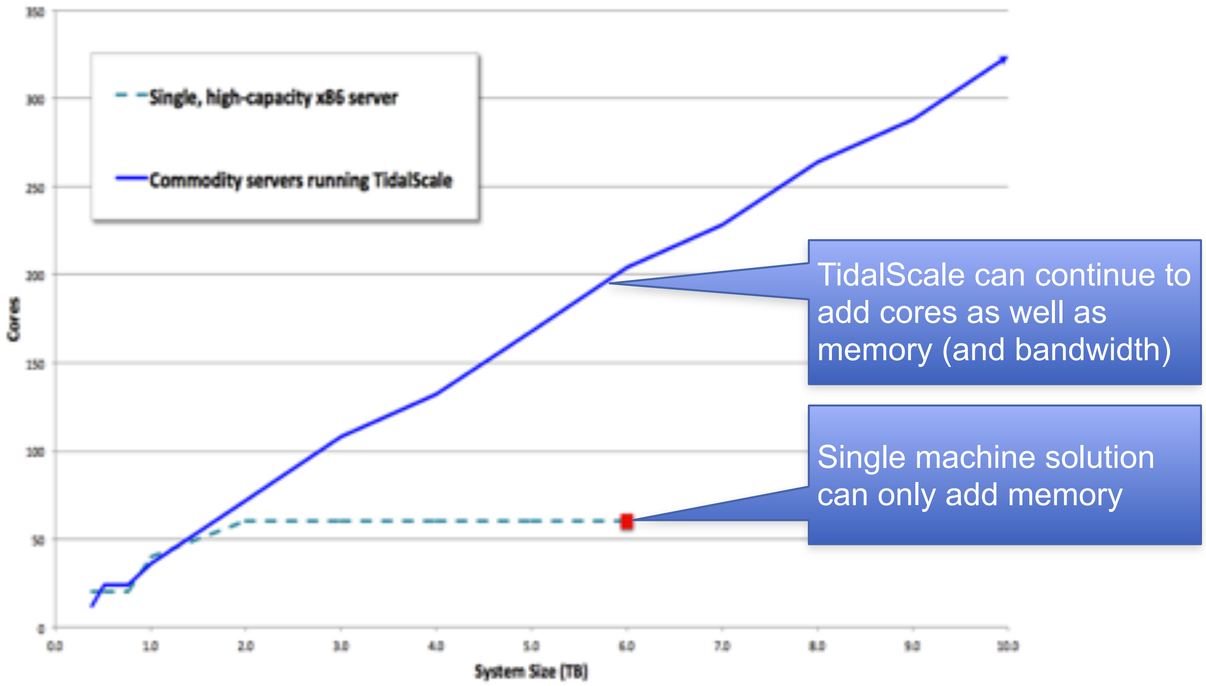
…and many other applications

- Industry Context
- Addressing the challenge
- A proposed solution
- **Review of the Benefits**

Price/Performance at Scale

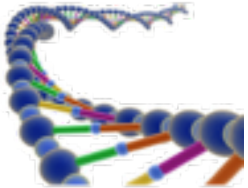


Performance Scales Up

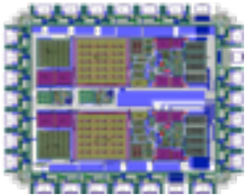


What Problems Benefit?

- **Data Mining / Finance**
  - High Data Volumes, Large Analytics, Risk Analysis, Fraud Detection, Graph Analytics using Alternative Data Sources, Risk modeling, High Frequency trading, Complex Event Modeling



- **Bioinformatics**
  - Next Generation DNA sequencing, Meta-genomic analysis, Finite Element Brain Modeling, Time-Series MRI Neuro-Imaging



- **IT / Operational Systems**
  - In-House Applications, Web Controllers & Servers, Gateways, Image serving, Ad serving, OLTP, ERP, Business Intelligence

Scale, Simplify, Optimize, Evolve

Scale:

- Aggregates compute resources for large scale in-memory analysis and decision support
- Scales like a cluster using commodity hardware at linear cost
- Allow customers to grow gradually as their needs develop

Simplify:

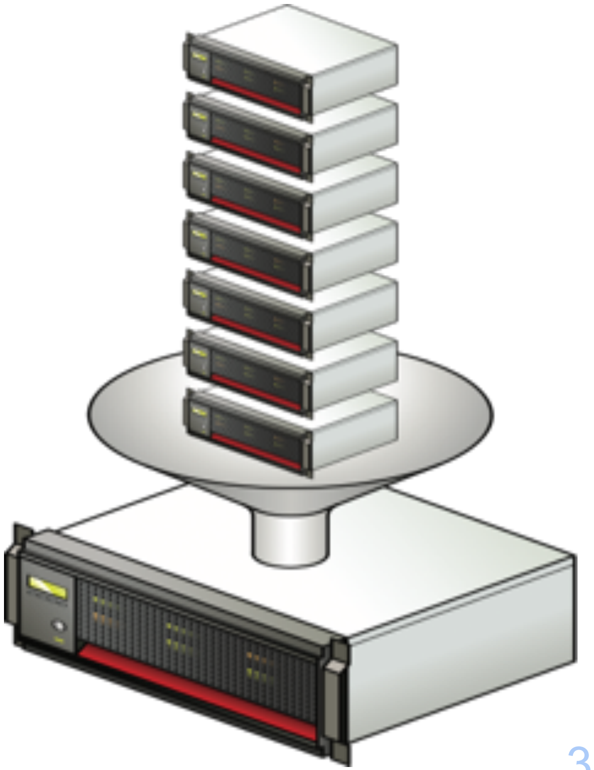
- Dramatically simplifies application development
- No need to distribute work across servers
- Existing applications run as a single instance, without modification, as if on a highly flexible mainframe

Optimize:

- Automatic dynamic hierarchical resource optimization

Evolve:

- Applicable to modern and emerging microprocessors, memories, interconnects, persistent storage & networks



Scale | Simplify | Optimize | Evolve

Restoring DEVELOPER PRODUCTIVITY THROUGH SIMPLICTY