

# How to Build a Software Defined Server, and How Best to Use it

---

Flexible - Fast - Easy

Ike Nassi  
Supercomputing Frontiers  
March 2017



# What if we think about a cluster differently?

- What if all the nodes in a cluster were combined to form a single virtual computer?
- What if that computer semi-automatically optimized itself?
  - That is, it exhibited *introspection* - watching itself, giving itself positive and negative feedback, and learning how to behave better?
  - What if the machine were better than we were about adjusting it's own behavior *through machine learning*?
  - And what if it did this at machine speed, rather than human speed?
- Could the computer be smarter than we are about managing its own operation?
- And what if it could get bigger and better without needing new generations of silicon?
- And how would we best make use of it?
  - E.g. through programming paradigms
  - What adjustment knobs would be useful?

# What is a Software Defined Server?

- *It is a collection of tightly-coupled cooperating servers, which together combine to form a single large computer running a single operating system, networked together on a standard interconnect.*
- The operating system should be standard, and not require *any* modifications.
- Applications should not require *any* modifications.
- Questions you might have:
  1. Can it be done at all?
  2. Isn't the interconnect going to be a bottleneck?
  3. Is there any special hardware required?
  4. Is it NUMA?
  5. Is it reliable?
  6. How do I set one up?

# TidalScale Snapshot

5

- Pioneering Software-Defined Servers
- Founded by Dr. Ike Nassi, CTO, in 2012
- Strong, cohesive team focused on revolutionizing the data center
- Software available for license **now**
- Backed by Bain Capital, HWVP, SAP Sapphire, Samsung, Citrix, InfoSys

**TidalScale**  
Software-Defined Servers

## Flexible

Scale the system to the size of the problem  
Use standard server hardware  
Grow the system as needs expand

## Fast

In-memory performance at large scale  
5TB, 10TB, 50TB of memory or more  
Dozens to hundreds of CPU cores

## Easy

Zero changes to the OS or applications  
Everything just works  
Transparent optimization via machine learning

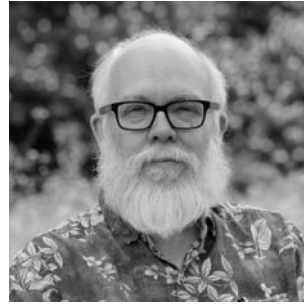
# An Experienced Team



**Dr. Ike Nassi**  
Founder / CTO



**Gary Smerdon**  
President & CEO



**Dr. David Reed**  
Chief Scientist



**Michael Berman**  
VP Engineering

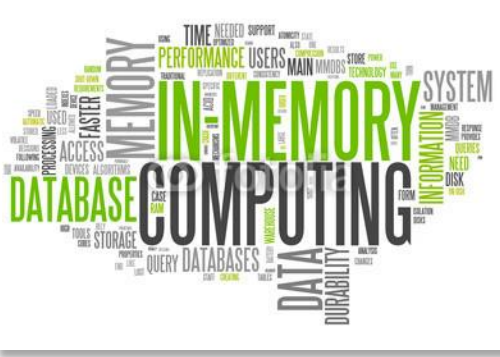


**Board / Advisors: Enrique Salem, Lars Leckie, Fred Weber, Carl Waldspurger, Gordon Bell**

# The Problem

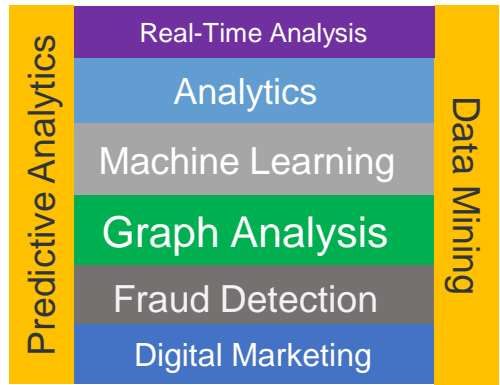
## High-Performance Demands

- The Challenge



Memory is 1000 times Faster than NVMe Flash!

## Data Explosion



Can my dataset fit in memory?

## Scale Up vs. Scale Out

	Scale Up	Scale Out
Software Simplicity	✓	✗
Hardware Cost	✗	✓

Do I have a choice?

What size of a system is needed in 3 years?

# Resolving the Conundrum of Scale Up or Scale Out

	Scale Up	Scale Out	TidalScale
Software Simplicity	✓	✗	✓
Hardware Cost	✗	✓	✓



# The Memory Hierarchy in Human Terms

Operation	Processing Latency
1 CPU Cycle	0.3 ns
L1 – L3 Cache	1 to 13 ns
DRAM	50 to 150 ns
Memory over Ethernet	3 $\mu$ s
CPU Context Transfer	6 $\mu$ s



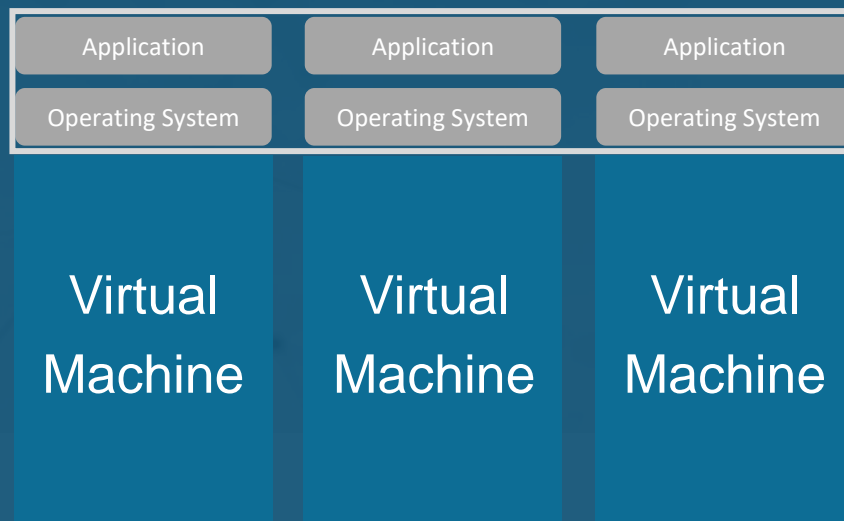
NVMe Flash	150 $\mu$ s
Flash Array	1 ms
TCP packet retransmit	2 s



# Technical Details

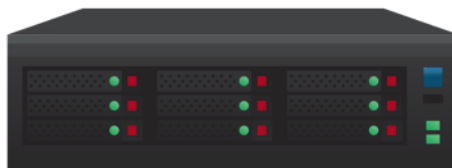
# Traditional Virtualization

Virtual



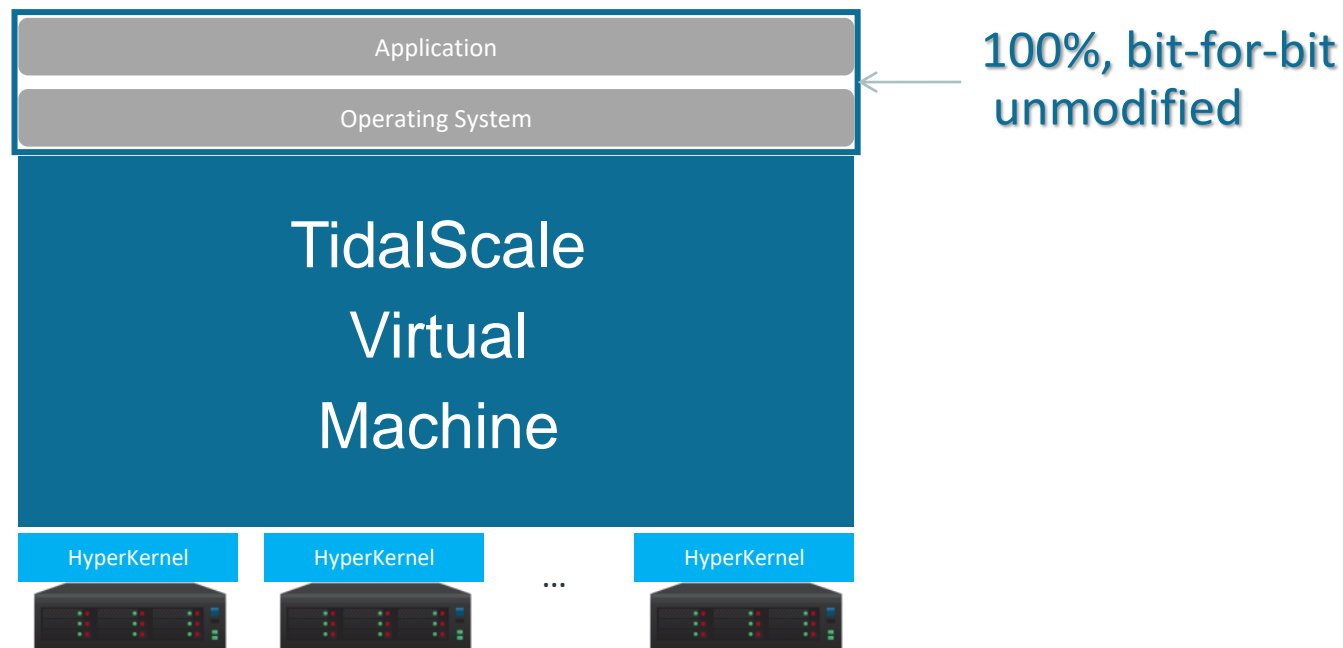
100%, bit-for-bit unmodified

Physical



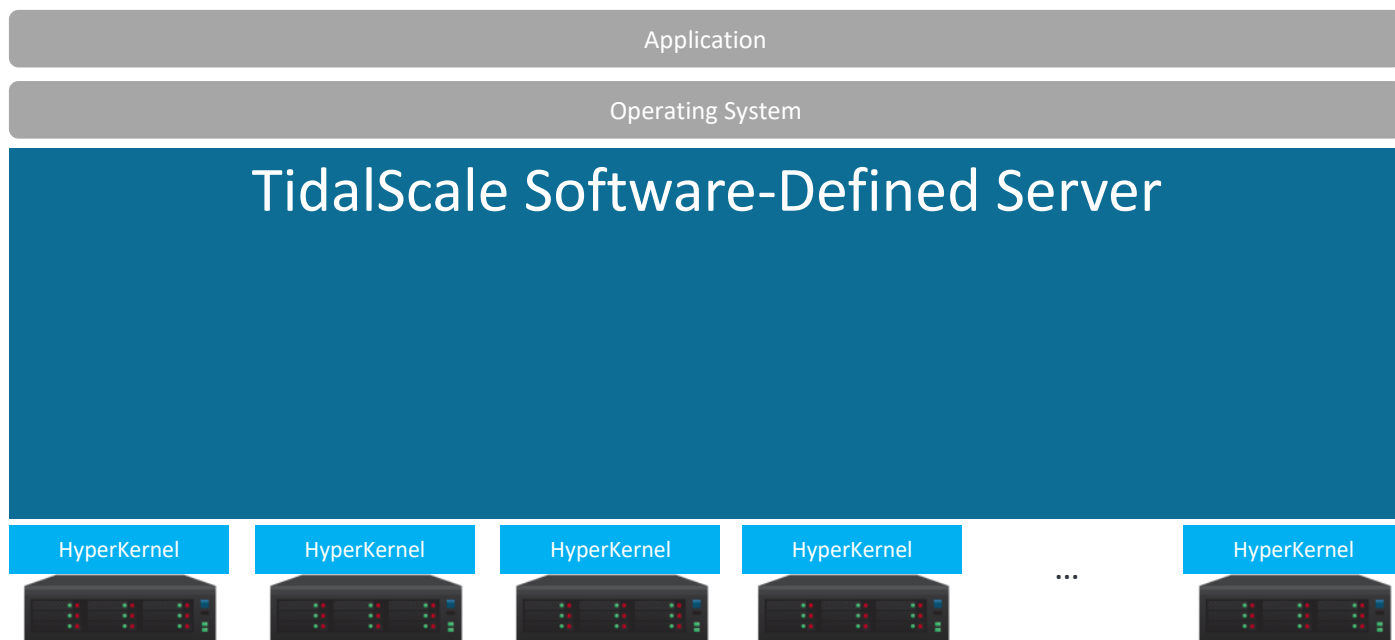
Multiple virtual machines share a single physical server

# TidalScale Software-Defined Servers



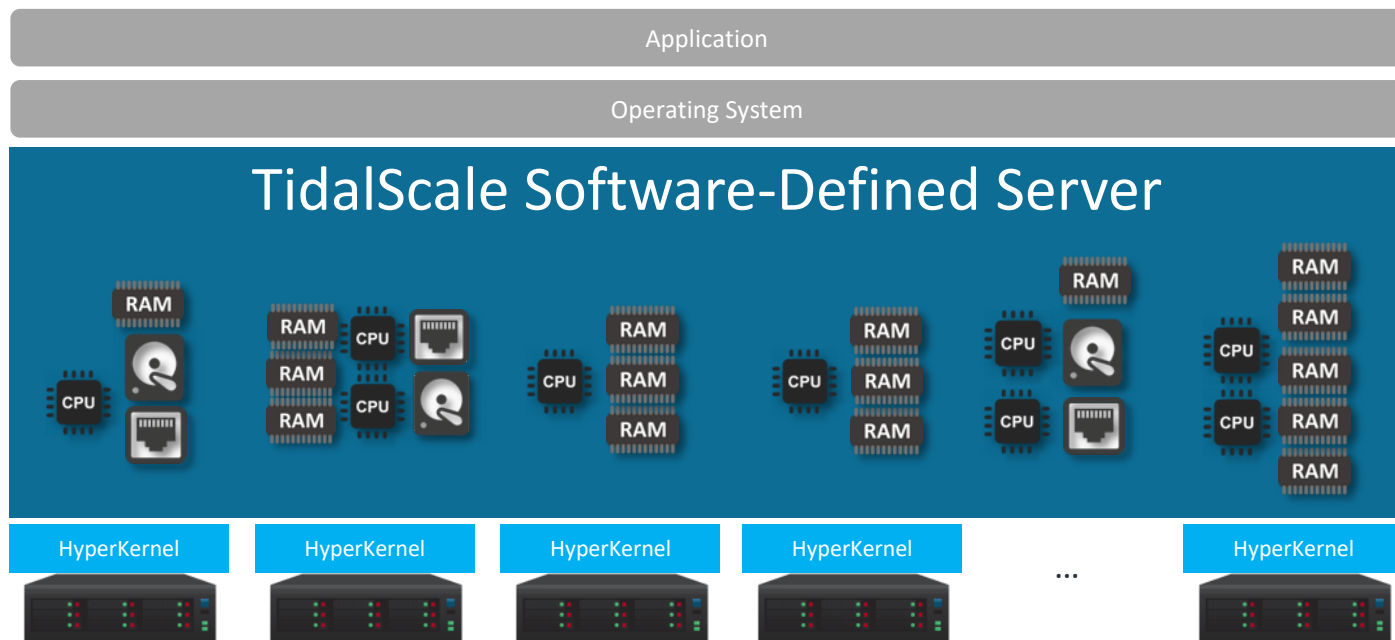
Single virtual machine spans multiple physical servers

# Seamless Scalability



Flexible – Scales Up or Down Quickly

# Machine Learning-Driven Self Optimization



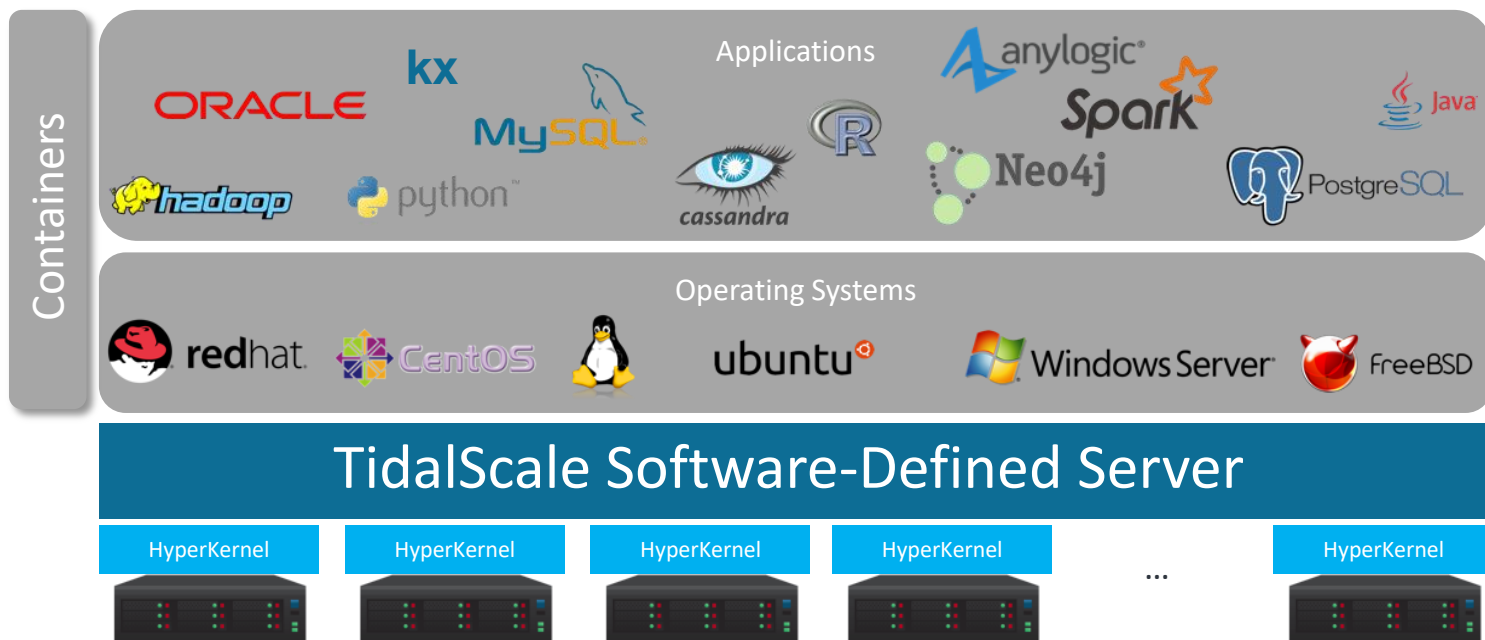
Uses patented machine learning to transparently align resources

# What a TidalScale HyperKernel Must Do

- Virtualize CPU, memory, I/O, and interrupts
- Mobilize all of these resources
- Decentralize all control – i.e. no shared Hyperkernel state, no central scheduler
- Be reliable
- Provide distributed, strongly coherent shared memory
- Preserve x86 execution order
- Boot unmodified guest OS and run unmodified software
- Scale linearly in cost, and dynamically over time
- Scale well as you add more nodes: memory bandwidth, PCI, etc.
- Perform well

**This enables customers to size the computer to their problem, rather than the other way around!**

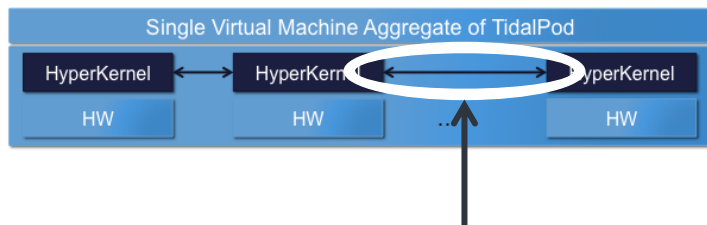
# 100% Compatible



If it runs, it runs on a TidalScale Software-Defined Server



# TidalScale Uses Ethernet as Resource Interconnect

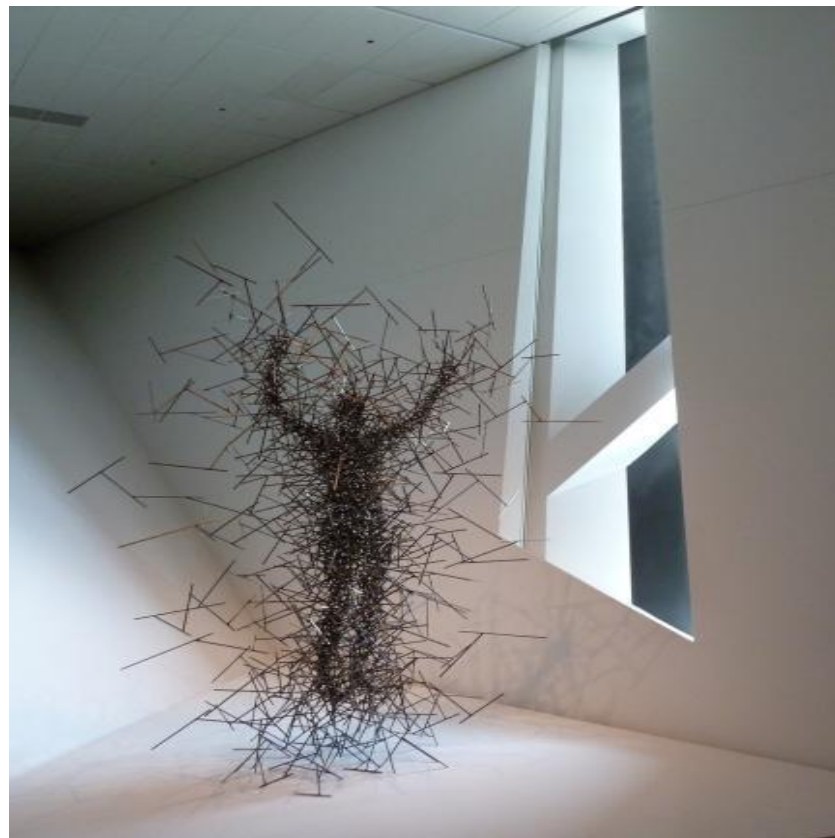


- 10 GBe switch or switch fabric (we use standard 10GE cards)
- High performance switch
- Private to HyperKernel, invisible to the OS
  - like a memory bus or I/O bus
- Low-latency, reliable, “zero-copy” Ethernet protocol
- 2 ports per node

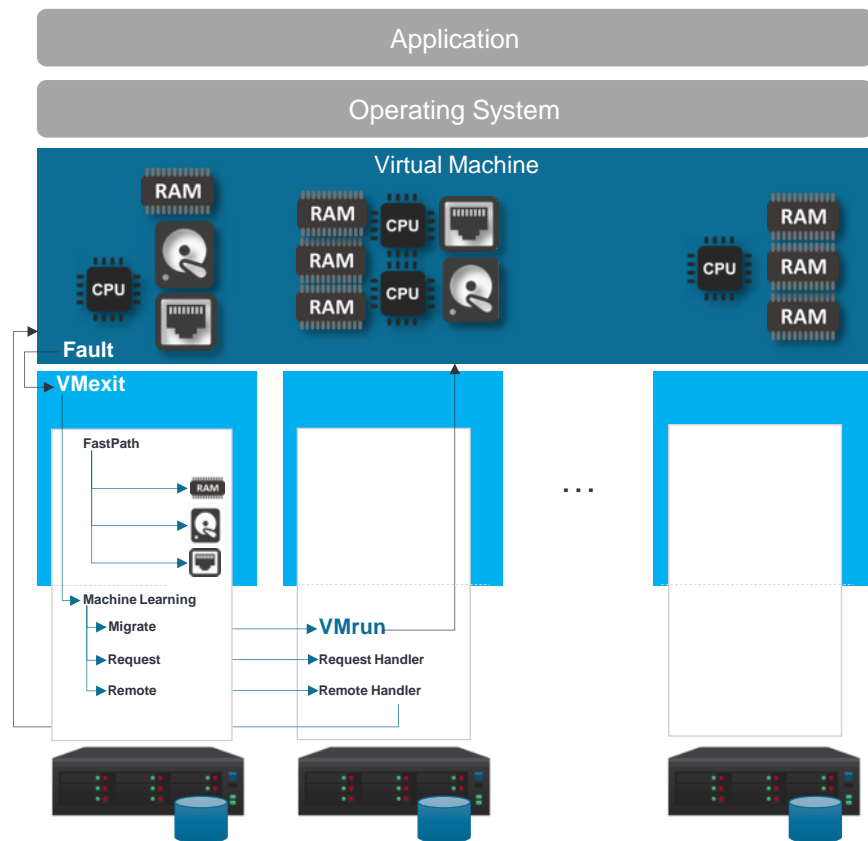
# Why It Works - Locality

- Nearly 50 years ago we figured out how to virtualize memory using the locality principle (i.e. working sets)\*
- Today, locality is applied ubiquitously across our computing infrastructure
- TidalScale applies locality to all compute resource types automatically & dynamically across physical machines
- We think in terms of memory hierarchies
- Think of DRAM as an L-4 cache of the VM

\* P.J. Denning

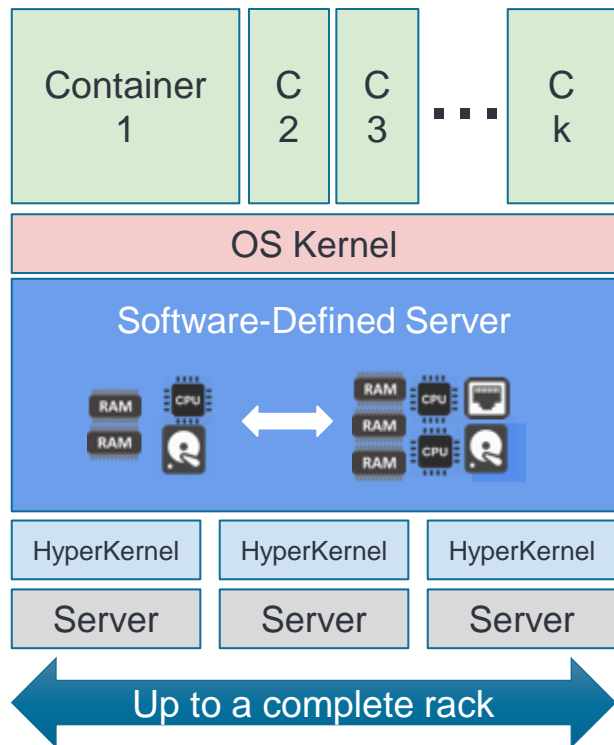


# HyperKernel ML Choices



- Resource Migration
  - Memory, Processor, I/O resources, and interrupts are moved across a resource interconnect to build and exploit locality.
- Remote Operation
  - Action transparently performed on remote node via resource interconnect.
- Replication
  - Selected resources are replicated to multiple nodes.
- Emulation
  - Native hardware operations can be emulated by the HyperKernel.

# Examples of IT Value: Containers



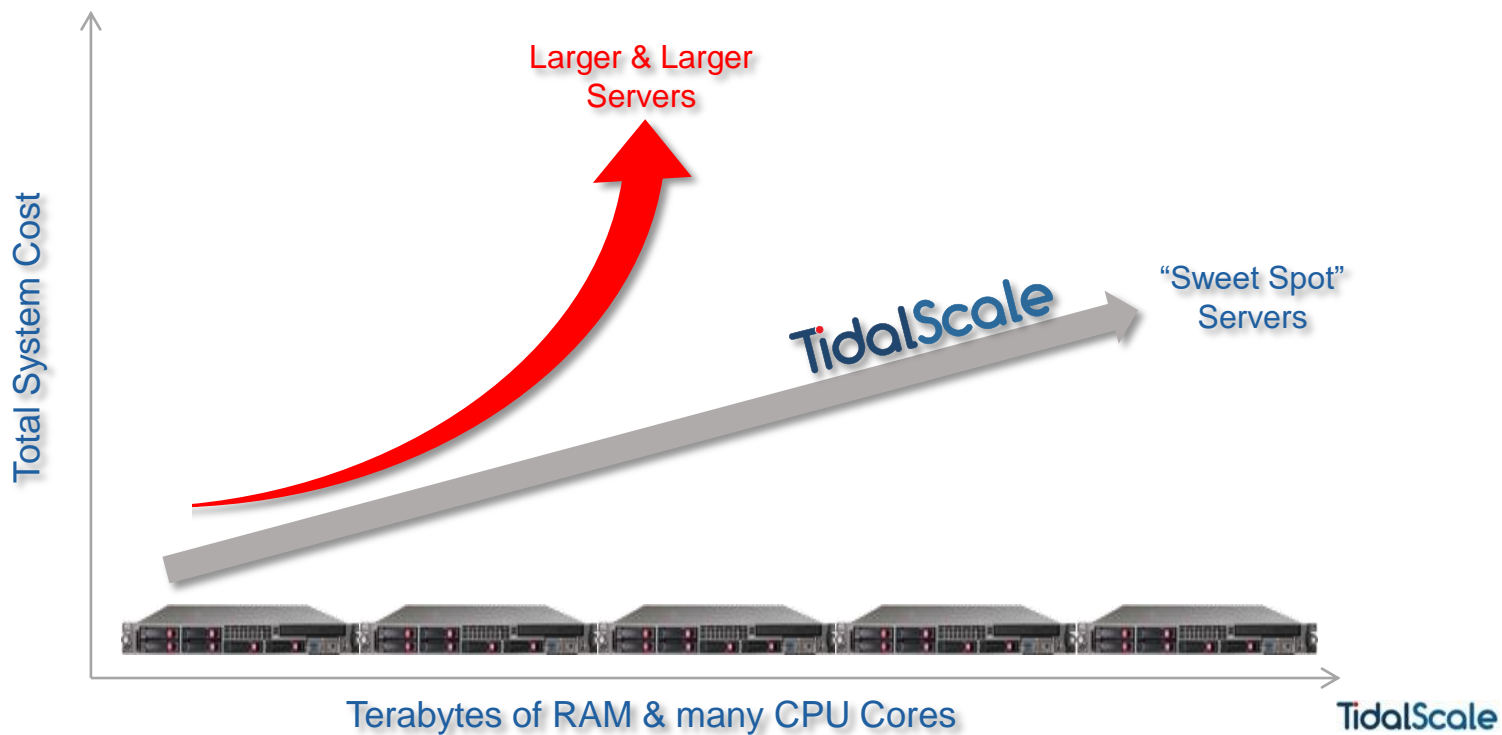
Multiple containers on a software-defined server means:

- Better bin packing
- High speed loopback networking
- Shared file system buffer cache
- Shared storage accruing all the benefits of a distributed file system *but* with the simplicity of a local filesystem.

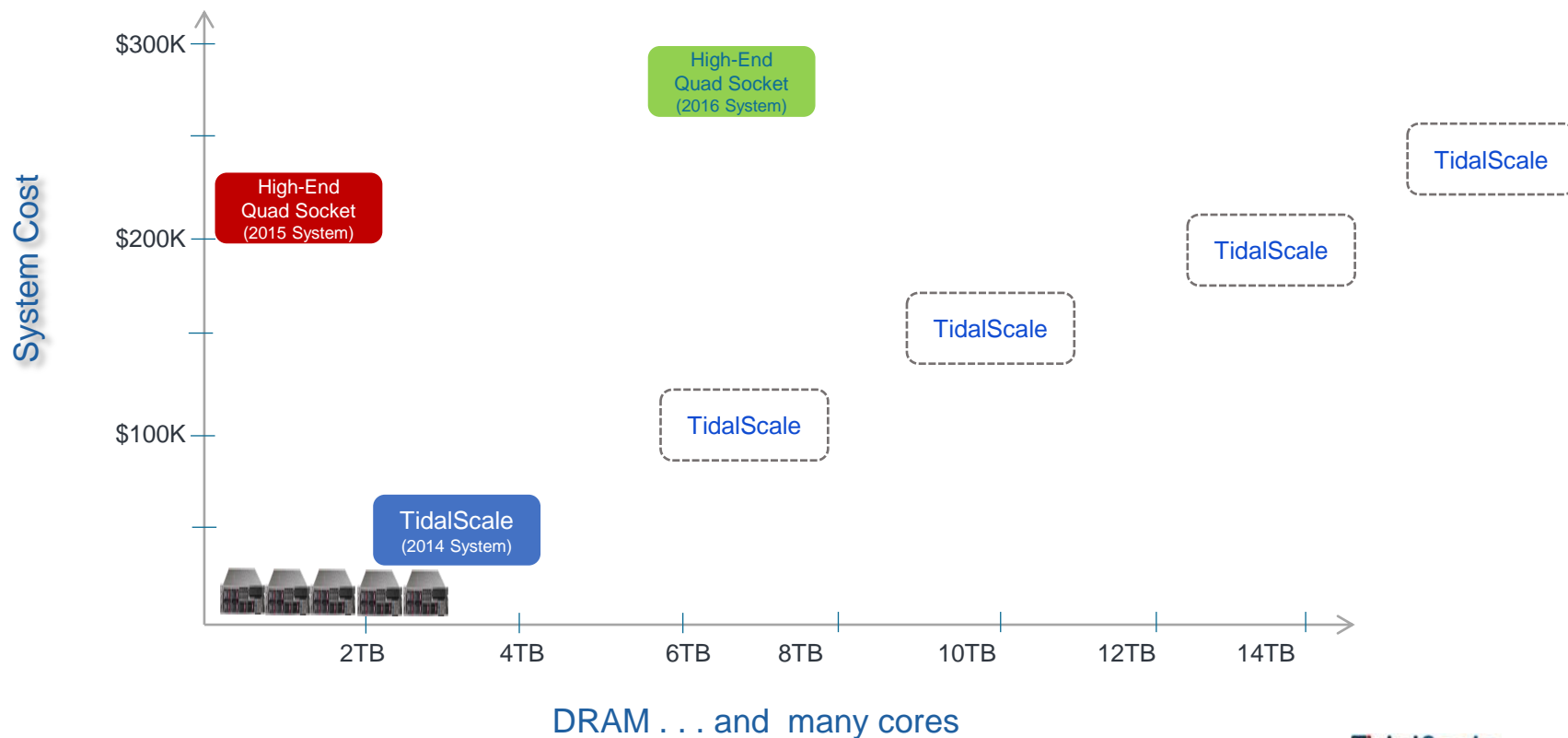


# Price and Performance

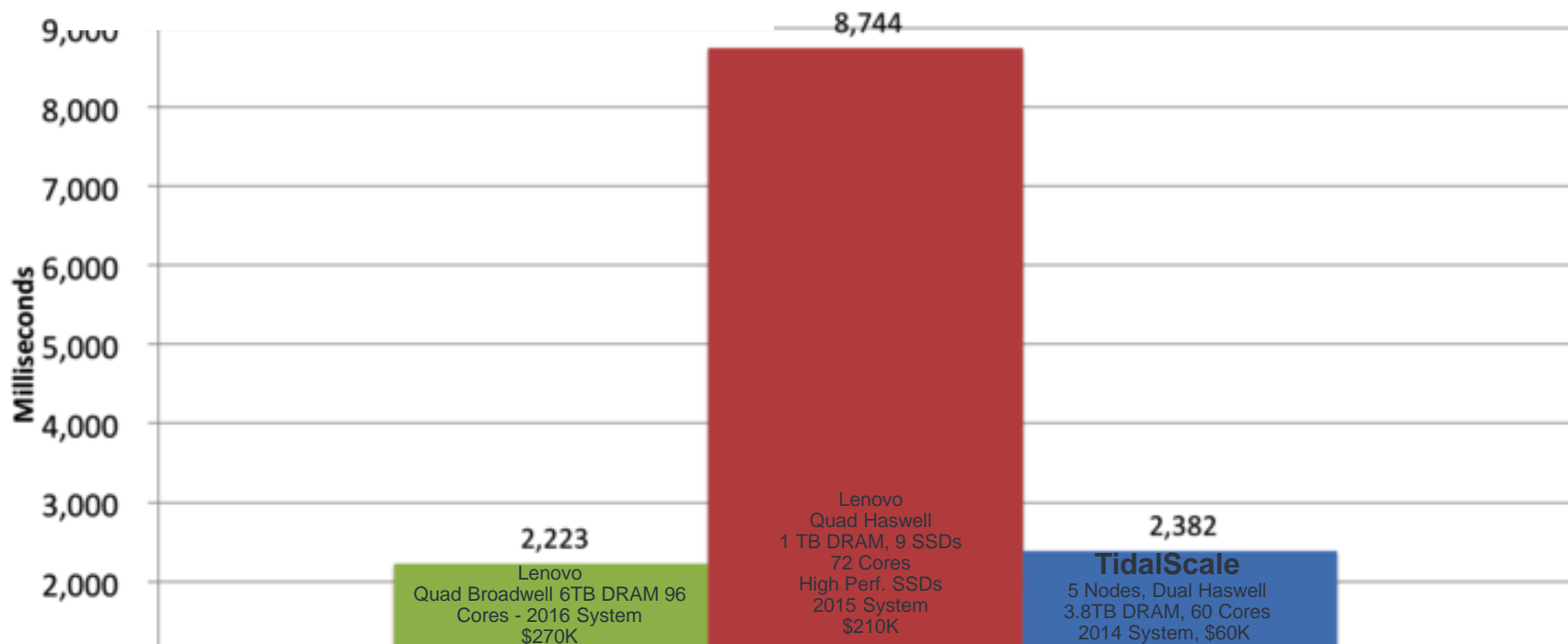
# Price/Performance at Scale



# Financial Analytic Price/Performance at Scale



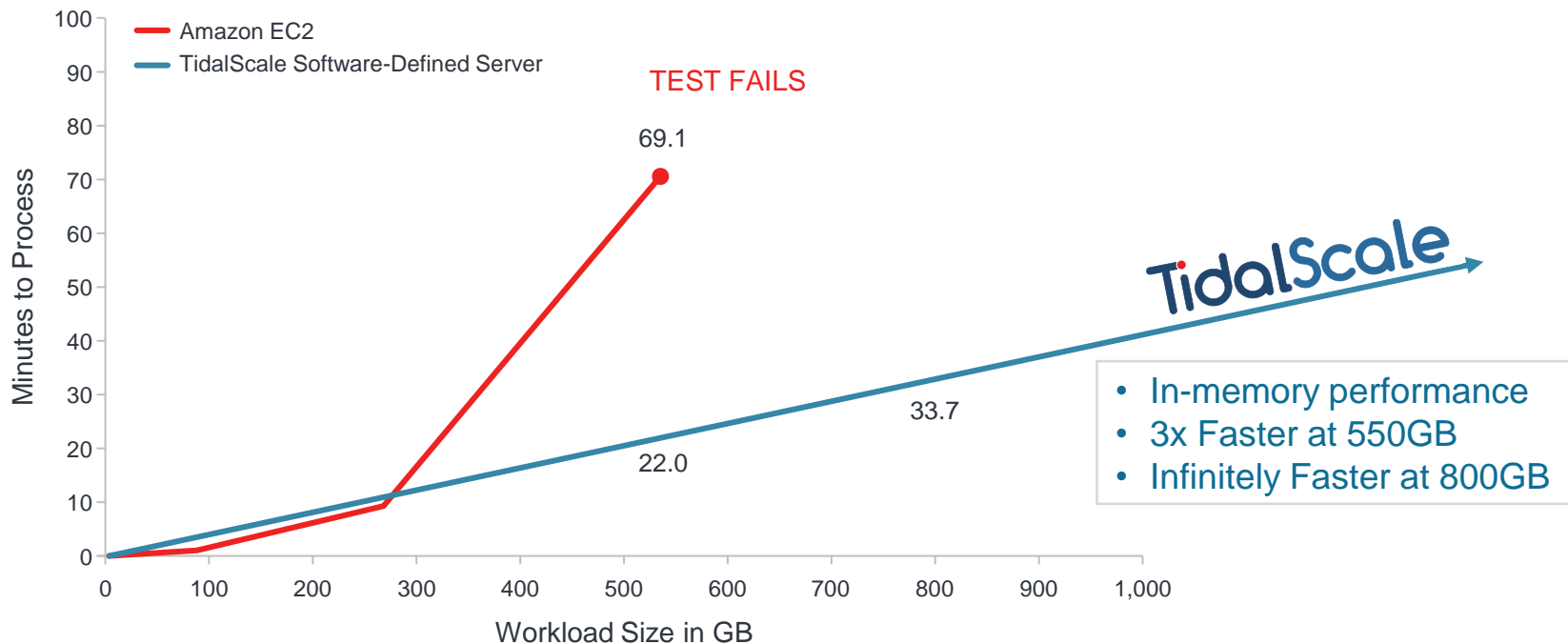
# Financial Analytic Test





# Retail Analytics on TidalScale

Performance Comparisons (TPC-H "Powertest" in Minutes)

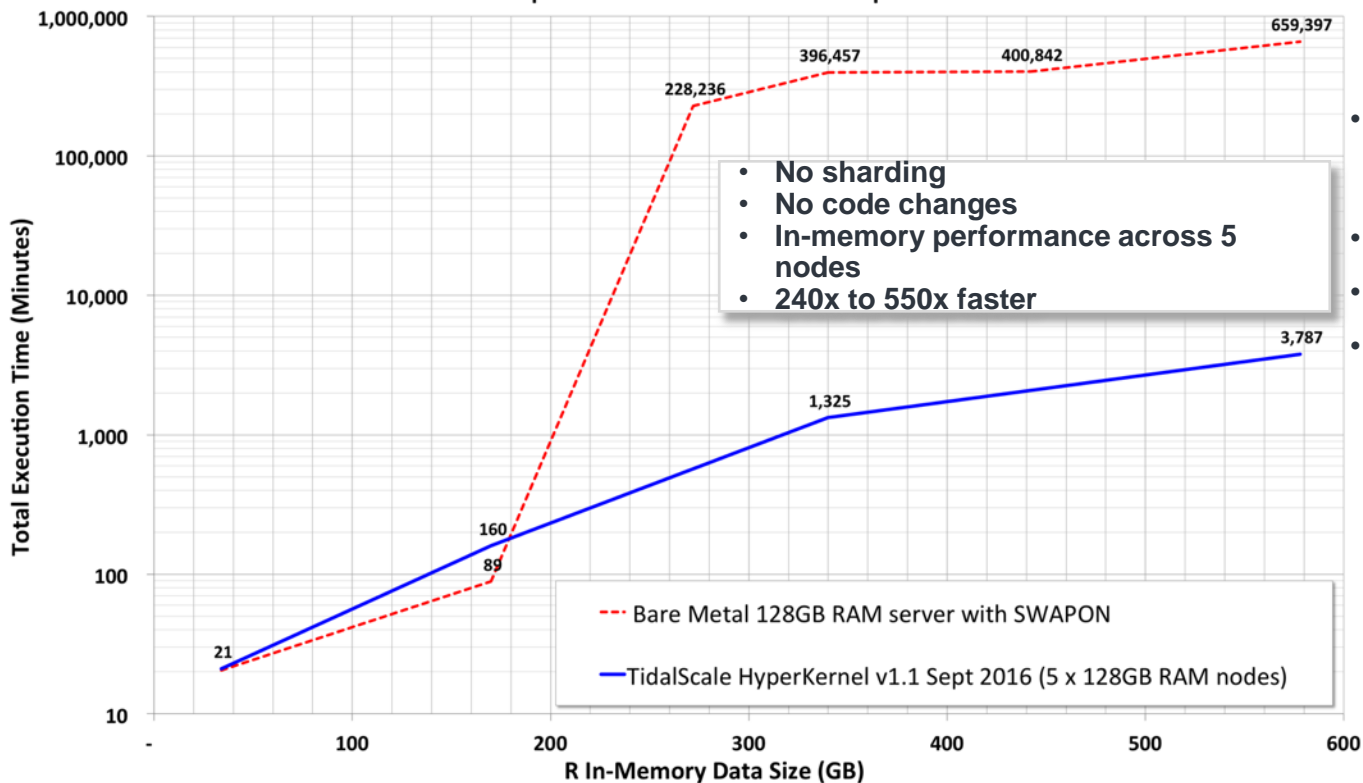


# Benchmark: Open Source R on TidalScale

<https://blog.tidalscale.com/300x-performance-gains-without-changing-a-line-of-code>

[https://github.com/TidalScale/R\\_benchmark\\_test](https://github.com/TidalScale/R_benchmark_test)

### Open Source R Performance Comparisons



- No sharding
- No code changes
- In-memory performance across 5 nodes
- 240x to 550x faster

--- Bare Metal 128GB RAM server with SWAPON  
— TidalScale HyperKernel v1.1 Sept 2016 (5 x 128GB RAM nodes)

- Version: Revolution R Open 8.0.3 with pryr, dplyr, mgcv, rpart, randomForest, FNN, Matrix, doparallel & foreach
- Data: CMS Public Use Dataset
- In-memory footprints: 32GB-680GB
- Operations timed:
  - Load
  - Join
  - GAM linear regression
  - GLM linear regression
  - Decision Tree
  - Random Forest (fixed seed)
  - K Nearest Neighbors





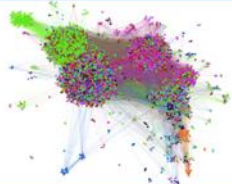
# Applications and Guidelines

# Application Use Cases

## Simulation, Modeling & Graph Analysis

### TidalScale Use Cases

### Simulation, Modeling, Graph Analysis



- Run fine-grained simulation models at large scale.
- Model entire populations and larger designs.
- Increase prediction accuracy.

Value	Scaling simulations takes memory and CPU resources. Scaling up increases the models capacity to handle accuracy, complexity, concurrency, and ultimately limit the exploration of the design space.
Applications	• Anylogic, Simulink

## Retail Analytics

### TidalScale Use Cases

### Retail Analytics



- Large volume, real time, retail analytics can enable analysts to improve
  - Retail Conversion Rate,
  - Average Purchase Value,
  - Items per Purchase and
  - Gross Margins.

Value	<ul style="list-style-type: none"> <li>• Improve supply chain visibility and forecast accuracy</li> <li>• Speed time to business insight and action</li> </ul>
Applications	• LogicBlox, Python, R, SAS

## Business Intelligence

### TidalScale Use Cases

### BI / OLAP



- Load and manipulate complete data sets at in-memory speed
- Get to insight faster on intra-day data streams.
- Ease real time processing of large data analytics.

Value	<ul style="list-style-type: none"> <li>• Asset/client aggregation: Merge tabs on clients and assets classes with unified view</li> <li>• Improved risk models: Aggregate risk information from more sources</li> <li>• Up-to-date forecasts: Integrate new data stream into predicting the future</li> </ul>
Applications	• SAP, Oracle RDBMS, Sybase, MS SQL, MySQL, IBM DB2

## Data Science on TidalScale

### TidalScale Use Cases

### High Productivity Languages



- Discover otherwise difficult to find patterns and relationships.
- Increase algorithm accuracy
- Deploy algorithms more quickly by avoiding application rewrite when moving to production on a scale out infrastructure.

Value	High productivity Languages enable Data scientists to quickly explore a dataset and rapidly change the methods they are leveraging to analyze the data space.
Applications	• Open Source R, Python, Matlab, SAS

TidalScale

## Financial Analytics

### TidalScale Use Cases

### Financial Analytics



- Ease processing of large volume financial analytics.
- Speed analysis of real time data streams.
- Operational flexibility enables IT to respond to rapidly changing analytic user needs.

Value	<ul style="list-style-type: none"> <li>• Run more simulations faster on trading strategies. Ensure compliance and evaluate execution systems performance.</li> </ul>
Applications	• Kdb+, Python, SAS

3/7/17

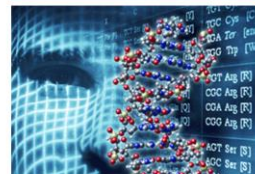
TidalScale Proprietary &amp; Confidential

28

## Bioinformatics

### TidalScale Use Cases

### Bioinformatics



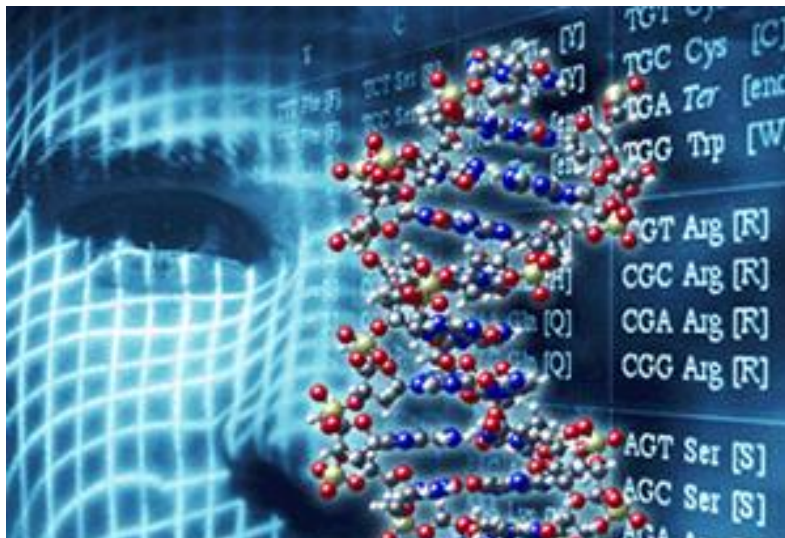
- Increase accuracy rates
- Ease processing of large genomic data sets
- Perform all analysis in-memory without intermediate storage steps
- Speed comparative analysis across populations
- Example: Analyze billions of genomic base pairs, differentially analyze, correlate differences with successful cancer treatment regimens, identify possible treatments

Value	<ul style="list-style-type: none"> <li>• Sequence larger organisms, identify organisms faster</li> <li>• Speed medical diagnosis and population-level genomic analysis for epidemiology</li> </ul>
Applications	• Galaxy, NGS algorithms (SOAPdenovo vs. SPAdes)

TidalScale

# Bioinformatics

## TidalScale Use Cases



## Bioinformatics

- Increase accuracy rates
- Ease processing of large genomic data sets
- Perform all analysis in-memory without intermediate storage steps
- Speed comparative analysis across populations
- Example: Analyze billions of genomic base pairs, differentially analyze, correlate differences with successful cancer treatment regimens, identify possible treatments

Value	<ul style="list-style-type: none"><li>• Sequence larger organisms, identify organisms faster</li><li>• Speed medical diagnosis and population-level genomic analysis for epidemiology</li></ul>
-------	---

Applications	<ul style="list-style-type: none"><li>• Galaxy, NGS algorithms (SOAPdenovo vs. SPAdes)</li></ul>
--------------	--

# HyperKernel Application Acceleration

- In-memory computing for SQL, NOSQL, Graph DBs, Hadoop, Spark,...
- Scale out R, SciPy, AnyLogic, without rearchitecture or API changes normally required for distributed computing
- Seamlessly manage velocity and volume without operational changes, i.e., scale Docker from 2 GB to 2 TB immediately
- ....

# HyperKernel In-Memory Computing Benefits

- Strictly speaking, manual sharding becomes unnecessary.
- Low-latency messages via Linux IPC (no TCP/IP).
- Work distribution via threads, instead of distributed computing.
- Build relationship networks in-memory, instead of across network, e.g., higher performing graph analytics.
- One programming model for the solution.

# How Applications can Leverage TidalScale

- TidalScale guests are an SSI that presents a large uniform memory architecture. Developers do not need to be concerned with data sharding or processor locality (usually).

Developers *should* be concerned with:

- In-memory computing and avoiding the memory cliff
- Algorithmic *inherent* non-locality
- Trading space for time
- False sharing



# Leveraging TidalScale: In-Memory Computing

- TidalScale exposes multiple TB of large distributed strongly coherent shared memory with uniform memory access.  
(Although we treat both as important, feedback from users suggests that memory is currently more important than cores.)
- Developers should:
  - Eliminate unnecessary I/O and convert storage objects into memory objects.
  - Stream data directly to memory.
  - Divide subtasks using Posix threads, OpenMP, or similar techniques.
  - Where possible, use pointers rather than move memory.
  - Avoid space-conserving algorithms that may be harmful to performance – they are unnecessary on TidalScale.

# Leveraging TidalScale: Avoid False Sharing

- TidalScale migrates pages for updates and replicates read-only/read-mostly pages.
- Developers should seek to avoid mixing local data of multiple thread's on the same page.
- Developers should allocate thread-local objects, use page-aligned memory, and allocate disjoint objects into unique pages.

# Example Financial Tech application

(based on a paper to appear in IEEE Computer)

- All trading data is historical
- 6,000 securities (e.g. AAPL, MSFT, INTC, etc.), 2 tables/security, 12,000 rows/table (including timestamp), 1,500 columns/table (~17TB)
- Be prepared to increase the dimensions
- For speed, keep it all in memory
- Problem:
  - Ingest securities data
  - Sort by timestamp
  - Respond to various queries
- Solution:
  - Create one thread per security
  - Each thread ingests it's own historical trading date in parallel
  - Create an array of several million *pointers to rows*
  - Sort *pointers to rows* by timestamp
  - No need to move any data – every core has direct addressability to every row

Blog, white paper, source code:

<https://blog.tidalscale.com/application-programming-when-memory-is-no-longer-a-constraint>

<https://www.tidalscale.com/hubfs/Marcom/White%20Papers/simple-shmem-current.c>



# Data Center Management

# Tomorrow Server's Today: A Game Changer



“Software-defined Servers make it easy to run memory-intensive applications like data mining, machine learning and simulation.”

Marc Jones, Director &  
Distinguished Engineer, IBM



# Create a Server of Any Size in Minutes



Step 1

Identify & import servers into a TidalPool



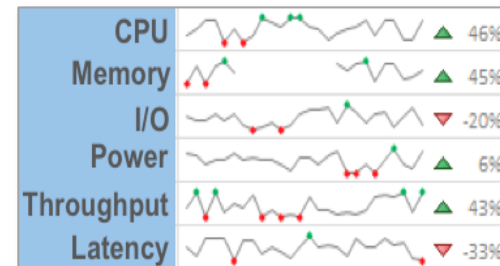
Step 2

Drag & Drop to create & manage **TidalPods**



Step 3

Monitor system performance & health



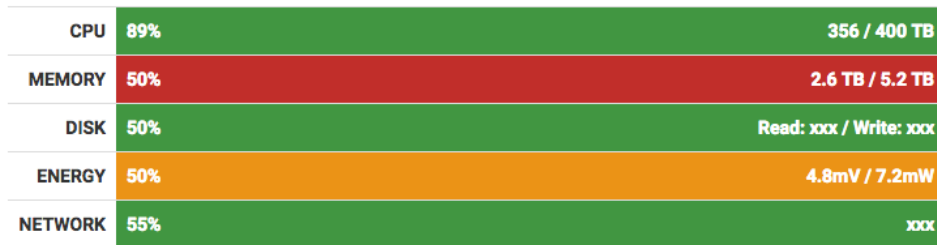
Deliver on *Flexible & Easy*

# Wave Runner – GUI Prototype

Tidal Pool: Tidal Pod 1: Engineering 1

TidalScale Waverunner 

SOCKETS: 40   CORES: 400   RAM: 63TB   STORAGE: 20TB   I/O: 800GB




Tidal Pod 1: WORKER NODES

NODE 1 

NODE 2 

 NODE 3 

CONTROLS 

LAYER	STATE	PROCEDURE
<input type="checkbox"/> ON <input type="checkbox"/> OFF GuestOS	Kernel Crash	xxx
<input type="checkbox"/> ON <input type="checkbox"/> OFF HyperKernel	Low Efficiency	xxx
<input type="checkbox"/> ON <input type="checkbox"/> OFF Worker Nodes	On	xxx

HOST:

CONFIGURE STORAGE

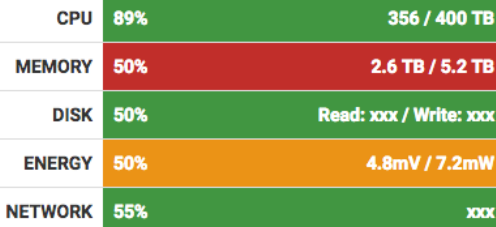
CONFIGURE NETWORK

HARDWARE VALIDATION

GUEST:

COMMAND LINE

CONSOLE



MIGRATE NODE

HYPERKERNEL:

VERSION   STATS   LOGS   DIAGNOSTICS   CONFIGURE

- Flexible: available in units of 512GB up to a full rack or more (15TB-23TB per rack)
- 1TB disk storage per unit
- 100MB Internet link
- Secure Access
- Firewall protection
- CentOS, Red Hat, Ubuntu,...
- Flexible lease options
- Persistent storage available

## Hosting Partners



IBM **Bluemix**<sup>™</sup>

**orion**<sup>™</sup> **VM**



# TidalScale – Development Systems

## Easy to Purchase

	“3+1” Starter Kit	“7+1” Starter Kit
Cores	48 Xeon E5v4 3.2Ghz	112 Xeon E5v4 3.2Ghz
Memory	1.5TB	3.6TB
Storage	7.2TB	10.4TB

## SuperMicro TwinMax Server



## Easy to Expand

Nodes	Cores	RAM
3 (current config)	48 Cores	1.5TB
7 Nodes	112 Cores	3.6TB
15 Nodes	240 Cores	7.7TB
23 Nodes	368 Cores	11.8TB

7.2TB	4
10.4TB	4
16.8TB	8
23.2TB	12

# Partnered with Industry Leaders



Infosys



TIBCO®



SIRQUL™

ubuntu®

orion™ VM



TidalScale

# Summary

- Why it's important to rethink some fundamental assumptions
- How it works
- How it performs
- Application coverage and some guidelines
- How to manage the landscape
- Deployment options

# Software-Defined Servers are a Game Changer

44

## Flexible

Scale to any size

Use commodity servers

Expand with user requirements

## Fast

In-memory performance

Optimize transparently with Machine Learning

Dispatch dozens to hundreds of CPU cores

## Easy

Run applications and OS unmodified

No partitioning datasets

Everything just works

**“This is the way all servers will be built in the future.”**

Gordon Bell, industry legend and 1<sup>st</sup> outside investor in TidalScale

TidalScale



Thank You

Ike Nassi  
ike.nassi@tidalscale.com

TidalScale

TidalScale