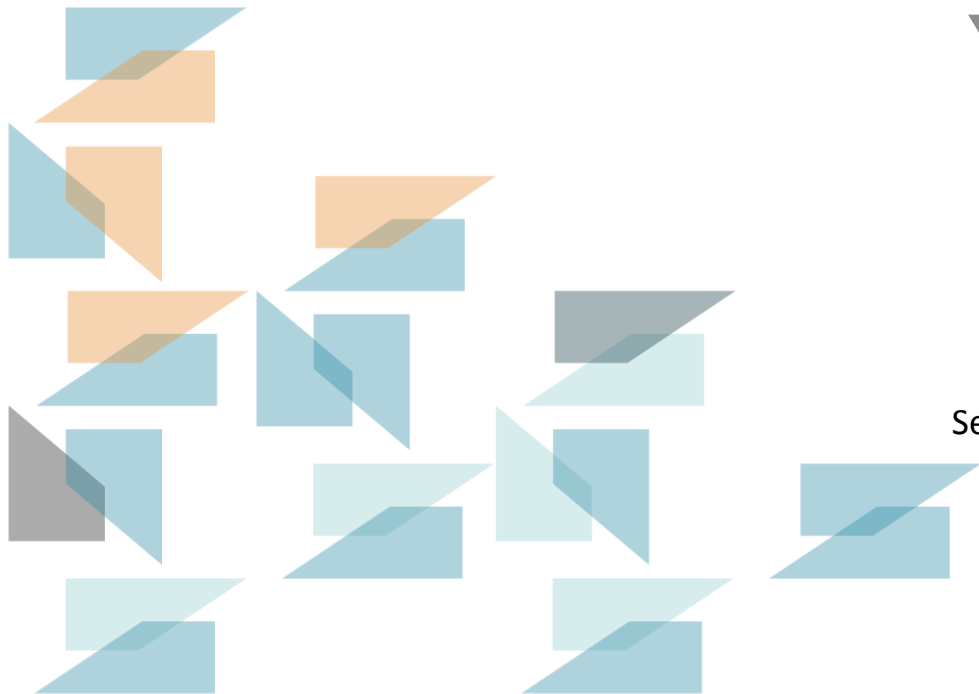


# Simulating large datasets using 2,3 TB of memory, 576 cores and 12 GPUs in one single Linux system

Scalable Solutions Designed to Grow with Your Needs



Atle Vesterkjær

Senior Software Engineer

[av@numascale.com](mailto:av@numascale.com)

February 2016

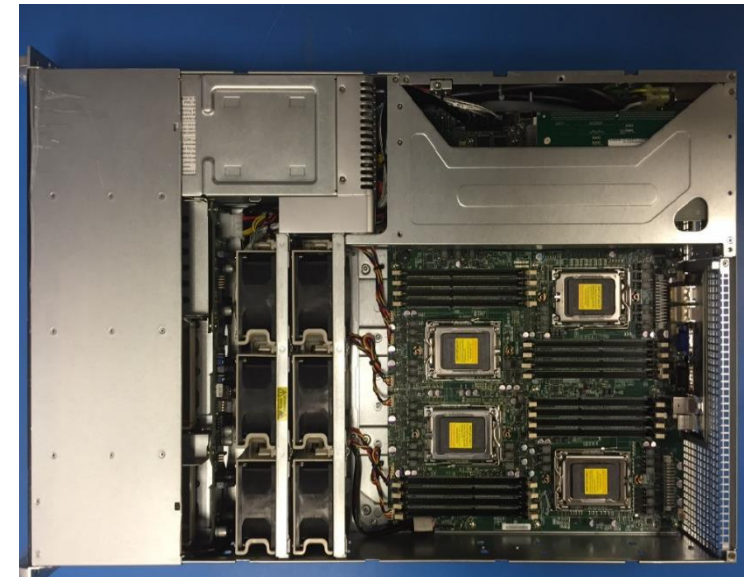
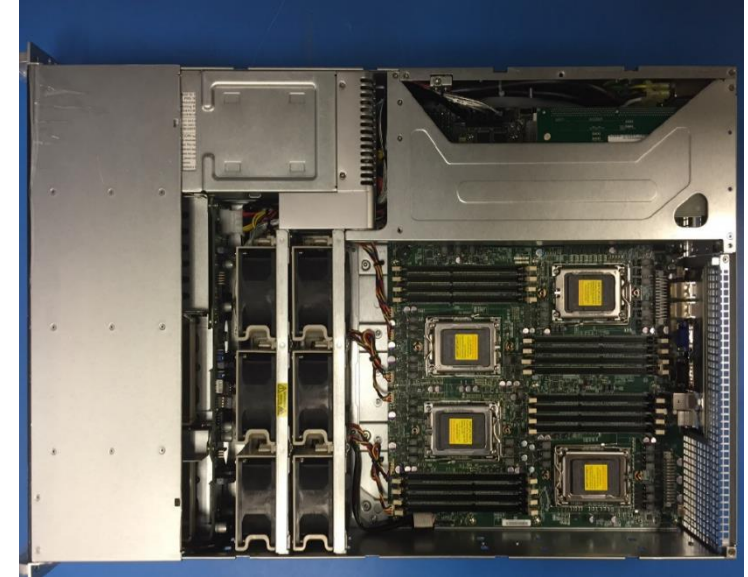
**NUMASCALE**  
BIGGER DATA ANALYTICS

# Need more cpus?



Traditional clustering solutions has unnatural boundaries in:

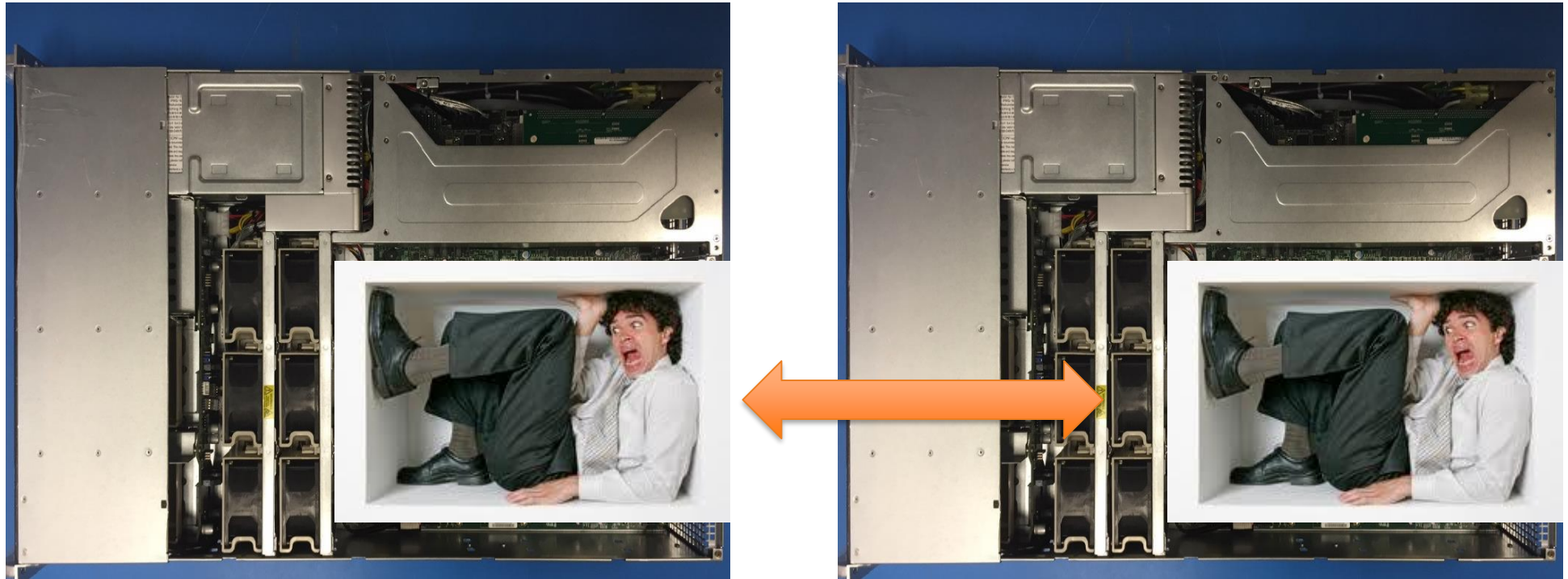
- RAM per server
- CPUs per server



# Out of memory space?

Traditional clustering solutions combine servers using

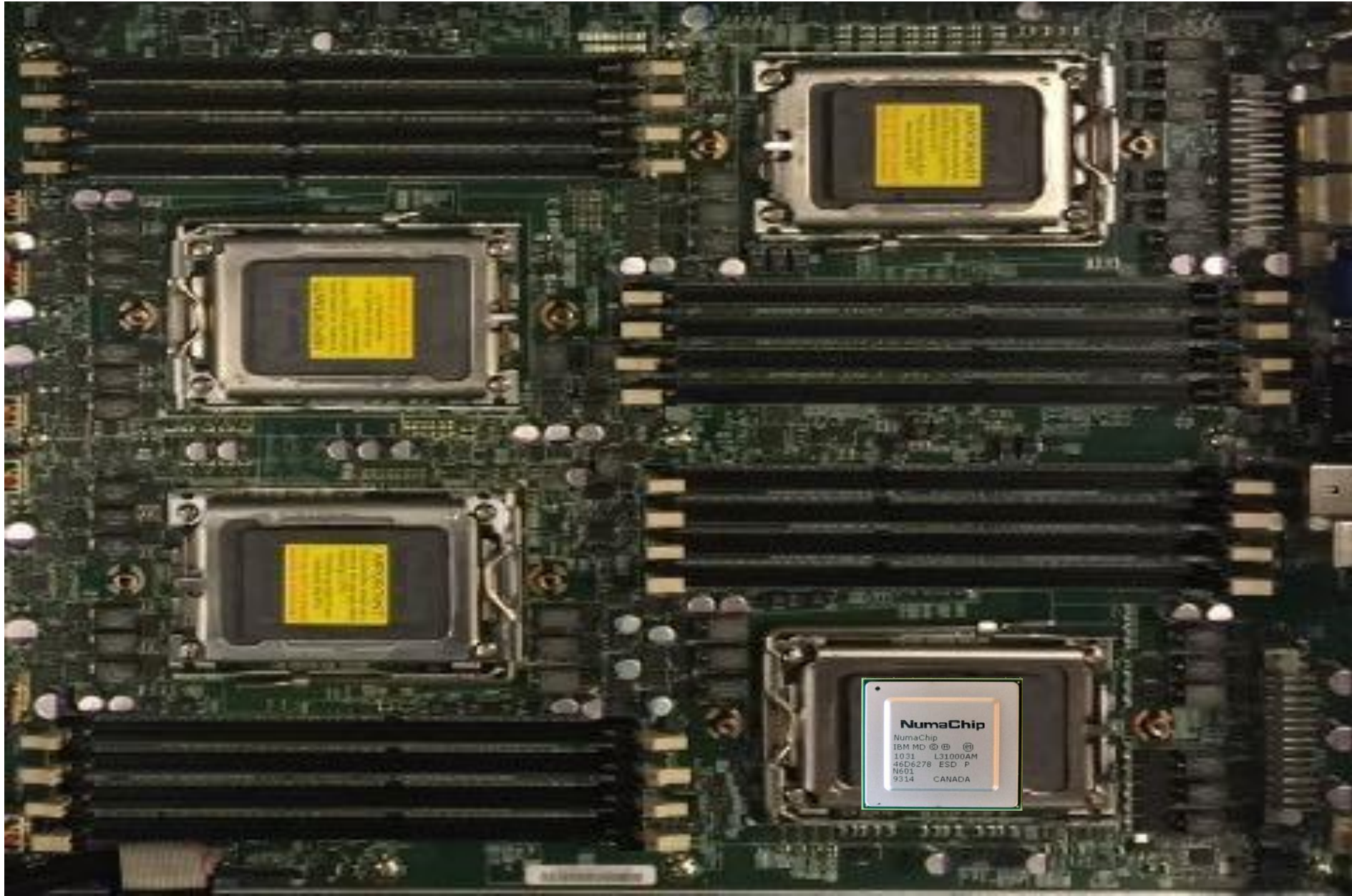
- Complex programming models
- IO adapters



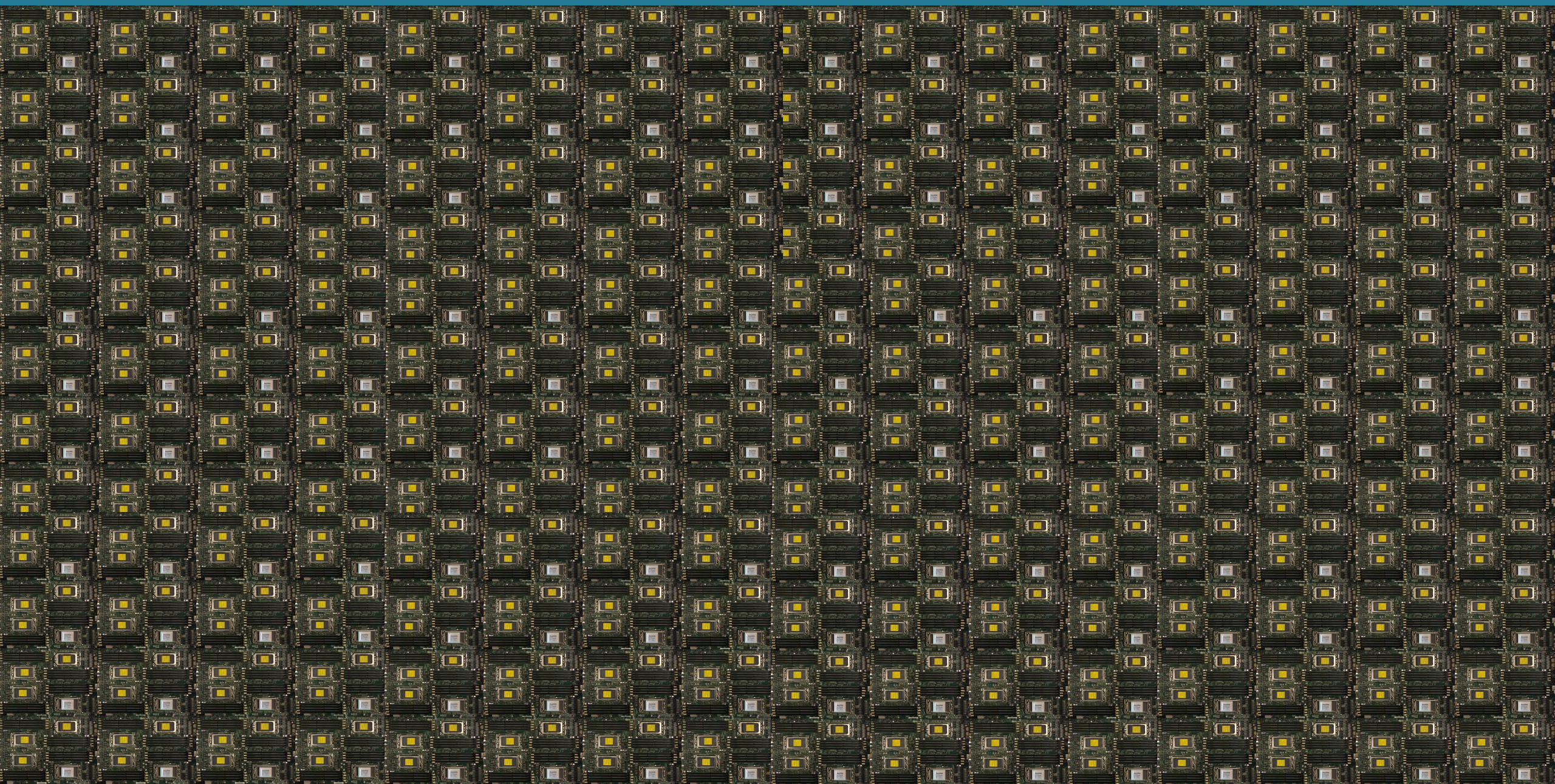
# Your system can grow organically with Numascale

Enourmous resources has  
been used to adapt to  
unnatural boundaries

It does not have to be this  
way



# Using the Native CPU communication protocol



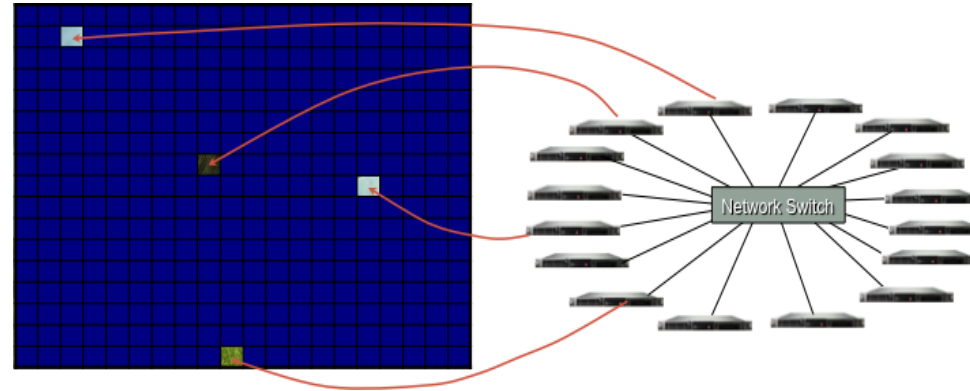
# Numascale's Unique Solution

## Numascale enables:

1. IN-MEMORY computing of large data sets at 100X higher speed by reducing interconnect latencies and avoiding disk paging = Efficient Predictive Analytics.
2. Powerful IO allows for real-time sensor and data input and rapid time to insights = Visualization.
3. Simplified software development, system utilization and reduced adm. and maintenance overhead by running single OS.
4. Low entry level cost and unmatched scalability

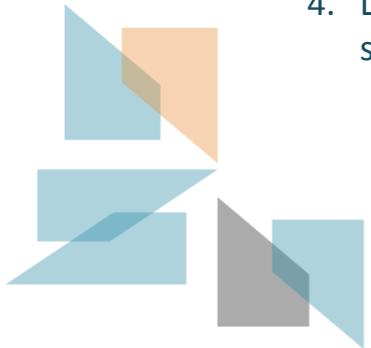
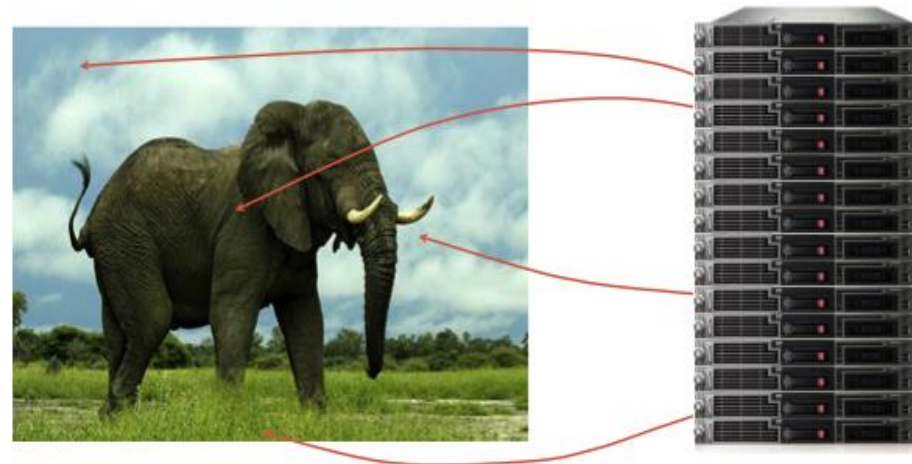
## Traditional Cluster:

Distributed memory gives partial views of data set



## Numascale "cluster":

Shared memory gives shared view of entire data set



# Unbeatable Programming Model

- Single System Image
- All applications and APIs
- NO Virtualization SW Layer!
- One System

```
top - 09:29:57 up 20 days, 20:07, 6 users, load average: 8.32, 8.31, 8.32
Tasks: 3068 total, 3 running, 3065 sleeping, 0 stopped, 0 zombie
Cpu(s): 1.4%us, 0.1%sy, 0.0%ni, 98.5%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 1583148160k total, 1185082348k used, 398065812k free, 4k buffers
Swap: 33046524k total, 0k used, 33046524k free, 4035576k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
202867	root	20	0	1119g	1.1t	524	R	815	74.1	3758:34	x.mod2as
197024	root	20	0	19160	6580	1204	R	37	0.0	403:40.55	htop
206397	root	20	0	13512	3704	924	R	14	0.0	0:04.48	top
197023	root	20	0	13512	3700	924	S	13	0.0	177:20.24	top
1399	root	20	0	0	0	0	S	3	0.0	1:03.79	ksoftirqd/348
2187	root	20	0	0	0	0	S	3	0.0	0:50.98	ksoftirqd/545
10	root	20	0	0	0	0	S	2	0.0	290:26.05	rcu_sched
11190	root	20	0	0	0	0	S	1	0.0	68:02.65	kworker/40:1
11177	root	20	0	0	0	0	S	1	0.0	173:47.85	kworker/24:1
11241	root	20	0	0	0	0	S	1	0.0	86:23.99	kworker/44:1
11668	root	20	0	0	0	0	S	1	0.0	53:37.71	kworker/348:1
11688	root	20	0	0	0	0	S	1	0.0	59:31.14	kworker/336:1
12035	root	20	0	0	0	0	S	1	0.0	20:37.15	kworker/545:1
11197	root	20	0	0	0	0	S	0	0.0	96:09.80	kworker/36:1
11203	root	20	0	0	0	0	S	0	0.0	148:15.71	kworker/32:1
11209	root	20	0	0	0	0	S	0	0.0	166:31.05	kworker/28:1
11233	root	20	0	0	0	0	S	0	0.0	82:54.52	kworker/48:1
11626	root	20	0	0	0	0	S	0	0.0	32:13.59	kworker/308:1
11710	root	20	0	0	0	0	S	0	0.0	26:04.65	kworker/357:1
11714	root	20	0	0	0	0	S	0	0.0	31:09.39	kworker/354:1
47717	root	20	0	7764	576	484	S	0	0.0	22:06.24	tail
47736	root	20	0	7764	576	484	S	0	0.0	22:13.57	tail



# Natural evolution of previous breakthroughs

- Step 1: Singlecore PC
- Step 2: Multicore PC
- Step 3: Multisocket Server
- Step 4: Scale out with bus bridging interconnect architecture
- Finally: Numascale systems share GPUs, CPUs, memory and IO between physical servers

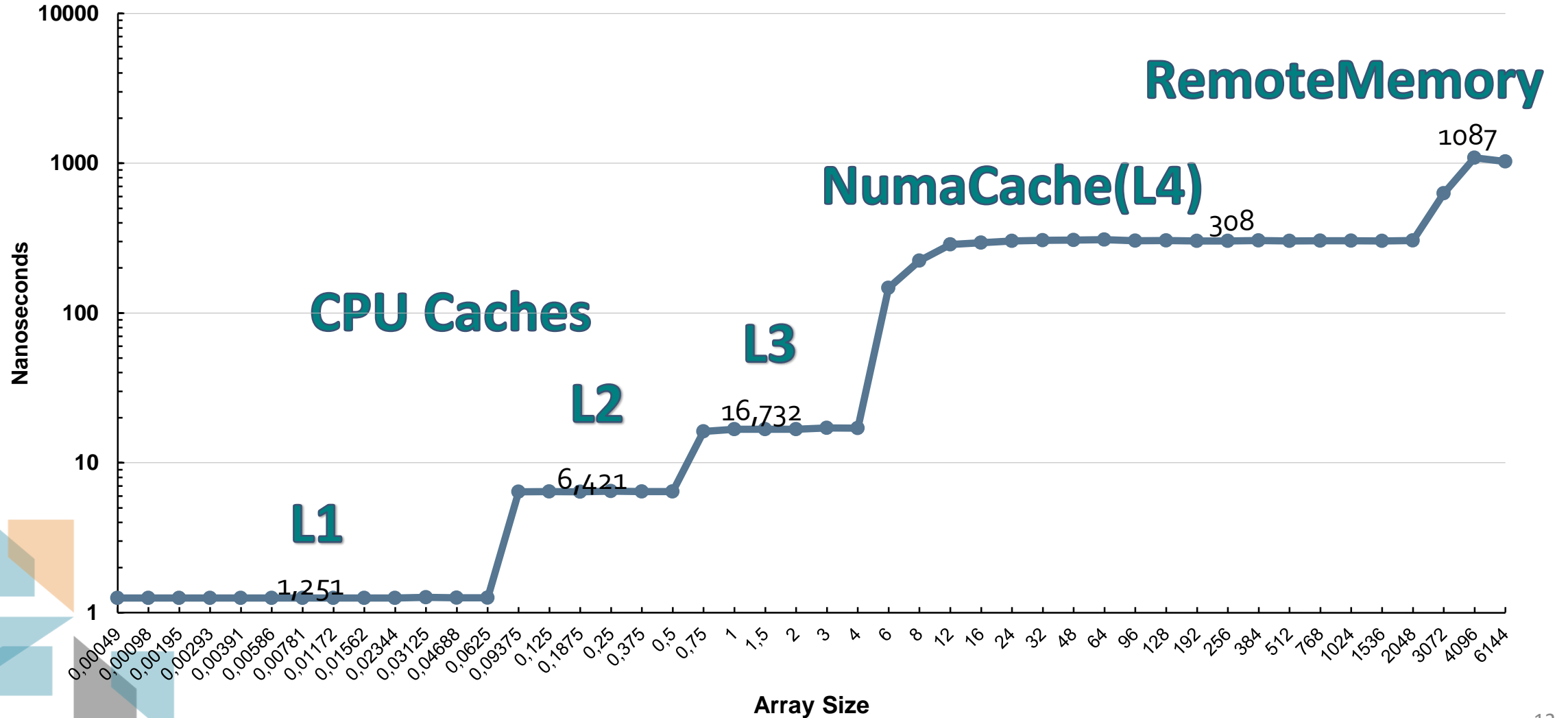


```
1 [|||||] 45.7% 37 [|||||] 28.1% 73 [|||||] 28.1% 109 [|||||] 28.6%
2 [|||||] 100.0% 38 [|||||] 100.0% 74 [|||||] 100.0% 110 [|||||] 100.0%
3 [|||||] 28.5% 39 [|||||] 28.1% 75 [|||||] 28.1% 111 [|||||] 28.5%
4 [|||||] 28.5% 40 [|||||] 28.0% 76 [|||||] 28.1% 112 [|||||] 28.6%
5 [|||||] 28.5% 41 [|||||] 28.0% 77 [|||||] 28.1% 113 [|||||] 28.6%
6 [|||||] 28.6% 42 [|||||] 28.1% 78 [|||||] 28.0% 114 [|||||] 28.6%
7 [|||||] 27.0% 43 [|||||] 27.5% 79 [|||||] 27.0% 115 [|||||] 27.5%
8 [|||||] 100.0% 44 [|||||] 100.0% 80 [|||||] 100.0% 116 [|||||] 53.3%
9 [|||||] 27.1% 45 [|||||] 27.5% 81 [|||||] 27.1% 117 [|||||] 27.5%
10 [|||||] 27.1% 46 [|||||] 27.5% 82 [|||||] 27.1% 118 [|||||] 100.0%
11 [|||||] 27.0% 47 [|||||] 27.5% 83 [|||||] 27.1% 119 [|||||] 27.6%
12 [|||||] 27.0% 48 [|||||] 27.5% 84 [|||||] 27.1% 120 [|||||] 27.5%
13 [|||||] 79.0% 49 [|||||] 77.5% 85 [|||||] 77.9% 121 [|||||] 79.0%
14 [|||||] 79.0% 50 [|||||] 100.0% 86 [|||||] 100.0% 122 [|||||] 100.0%
15 [|||||] 79.0% 51 [|||||] 77.9% 87 [|||||] 77.9% 123 [|||||] 79.0%
16 [|||||] 79.0% 52 [|||||] 77.5% 88 [|||||] 77.5% 124 [|||||] 79.1%
17 [|||||] 79.0% 53 [|||||] 77.5% 89 [|||||] 77.5% 125 [|||||] 79.1%
18 [|||||] 100.0% 54 [|||||] 77.5% 90 [|||||] 77.5% 126 [|||||] 79.1%
19 [|||||] 61.0% 55 [|||||] 62.0% 91 [|||||] 62.3% 127 [|||||] 61.2%
20 [|||||] 100.0% 56 [|||||] 62.0% 92 [|||||] 100.0% 128 [|||||] 61.2%
21 [|||||] 61.3% 57 [|||||] 100.0% 93 [|||||] 62.0% 129 [|||||] 61.2%
22 [|||||] 61.0% 58 [|||||] 62.0% 94 [|||||] 62.3% 130 [|||||] 100.0%
23 [|||||] 61.3% 59 [|||||] 62.0% 95 [|||||] 62.3% 131 [|||||] 61.2%
24 [|||||] 61.0% 60 [|||||] 62.0% 96 [|||||] 62.0% 132 [|||||] 61.2%
25 [|||||] 57.5% 61 [|||||] 60.0% 97 [|||||] 60.5% 133 [|||||] 57.2%
26 [|||||] 100.0% 62 [|||||] 100.0% 98 [|||||] 60.5% 134 [|||||] 57.2%
27 [|||||] 57.5% 63 [|||||] 60.0% 99 [|||||] 60.5% 135 [|||||] 100.0%
28 [|||||] 57.5% 64 [|||||] 60.0% 100 [|||||] 60.5% 136 [|||||] 57.2%
29 [|||||] 57.5% 65 [|||||] 60.0% 101 [|||||] 60.5% 137 [|||||] 57.5%
30 [|||||] 57.5% 66 [|||||] 60.3% 102 [|||||] 100.0% 138 [|||||] 57.2%
31 [|||||] 37.0% 67 [|||||] 37.2% 103 [|||||] 38.0% 139 [|||||] 36.8%
32 [|||||] 100.0% 68 [|||||] 100.0% 104 [|||||] 100.0% 140 [|||||] 100.0%
33 [|||||] 37.0% 69 [|||||] 37.2% 105 [|||||] 38.0% 141 [|||||] 36.8%
34 [|||||] 37.0% 70 [|||||] 37.2% 106 [|||||] 38.0% 142 [|||||] 36.8%
35 [|||||] 37.0% 71 [|||||] 37.2% 107 [|||||] 38.0% 143 [|||||] 36.6%
36 [|||||] 37.2% 72 [|||||] 37.2% 108 [|||||] 38.0% 144 [|||||] 36.8%
Mem [|||||] 52118/384880MB Tasks: 54, 130 thr; 144 running
Swp [|||||] 0/0MB Load average: 49.64 13.40 4.74
Uptime: 00:05:35

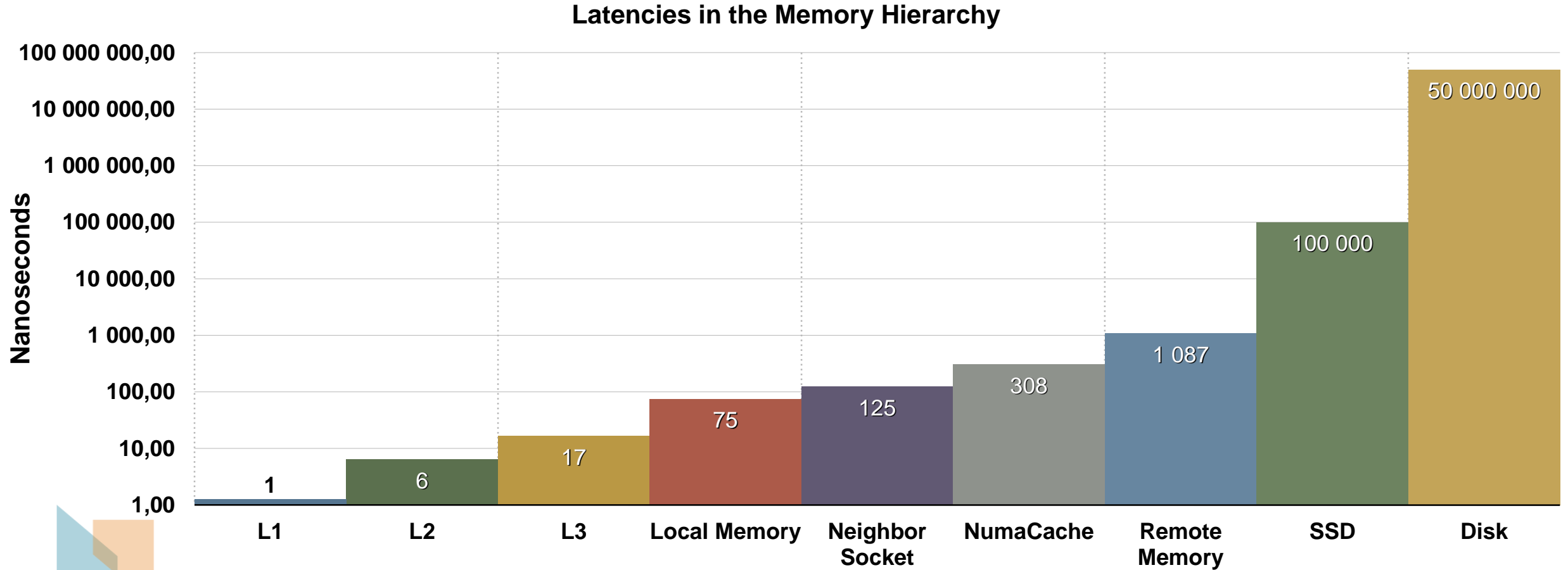
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
10635 root 20 0 2600M 2117M 4128 R 636. 0.6 2:55.09 Linux_AMD64_ACML440_MP/xhpl /tmp/HPL.dat.zk0RhF
10623 root 20 0 2613M 2102M 4156 R 636. 0.5 2:50.99 Linux_AMD64_ACML440_MP/xhpl /tmp/HPL.dat.zk0RhF
10629 root 20 0 2613M 2115M 4124 R 634. 0.5 2:52.77 Linux_AMD64_ACML440_MP/xhpl /tmp/HPL.dat.zk0RhF
10641 root 20 0 2588M 2094M 4124 R 633. 0.5 2:57.50 Linux_AMD64_ACML440_MP/xhpl /tmp/HPL.dat.zk0RhF
10636 root 20 0 2627M 2129M 4152 R 561. 0.6 2:51.58 Linux_AMD64_ACML440_MP/xhpl /tmp/HPL.dat.zk0RhF
10642 root 20 0 2602M 2112M 4168 R 557. 0.5 2:56.39 Linux_AMD64_ACML440_MP/xhpl /tmp/HPL.dat.zk0RhF
10630 root 20 0 2627M 2130M 4180 R 556. 0.6 2:51.86 Linux_AMD64_ACML440_MP/xhpl /tmp/HPL.dat.zk0RhF
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice F8Nice + F9Kill F10Quit
```

# LMbench - Chart

Latency - LMBench



# The Memory Hierarchy



# Braking records with Open Source Software

- Numascale Appliances comes with **Open Source Software Stack**.
- Numascale builds larger single image systems than anyone else.
- Patches to the Linux kernel, libraries and tools **returned to the open source community**



## Numascale Wiki

Home **Tips** Recommended configurations Recipes Environment

10 ▾

Title	Author	Hits
<a href="#">Performance/stability debugging</a>	Written by Daniel J Blueman	Hits: 33
<a href="#">Numascale Best Practice Guide</a>	Written by Atle Vesterkjær	Hits: 96
<a href="#">Building thread-safe ScaLAPACK</a>	Written by Stefan Olbrich	Hits: 501
<a href="#">Numaplace</a>	Written by Stefan Olbrich	Hits: 398
<a href="#">The NCALLOC memory allocator</a>	Written by Stefan Olbrich	Hits: 152
<a href="#">Kernel patches</a>	Written by Daniel J Blueman	Hits: 185
<a href="#">OpenMPI using NumaConnect BTL</a>	Written by Stefan Olbrich	Hits: 296
<a href="#">OS tips</a>	Written by Daniel J Blueman	Hits: 320
<a href="#">Run-time tips</a>	Written by Daniel J Blueman	Hits: 267
<a href="#">Compiler tips</a>	Written by Daniel J Blueman	Hits: 326

© 2015 Numascale Wiki [Back to Top](#)

# Easier Parallel Programming

- Any **multithreaded** application can run on NumaConnect using e.g pthreads.
- **OpenMP** can be used for easily applying a parallel programming model for a shared memory computer and will scale the execution of parallel regions in an application across many threads
- Numascale develops preloaded modules for R, **OpenMPI**, **LAPACK**, **memory allocation and more** that fully utilize the NumaConnect



# Worlds Largest SMP

**108 nodes**  
**324 AMD 6386 CPUs**  
**5184 cores**  
**20.7TBytes Memory**

*Application: Real time Big  
Data Analytics*



# World Record Memory Bandwidth

-----  
STREAM version \$Revision: 5.10 \$  
-----

This system uses 8 bytes per array element.  
-----

Array size = 135000000000 (elements), Offset = 0  
Total memory required = 3089904.8 MiB (= 3017.5 GiB).  
Each test is run 10 times, but only  
the \*best\* time for each is used.  
-----

Number of Threads requested = 1296  
-----

Function	Best Rate MB/s	Avg time	Min time	Max time
Copy:	9139226.2	0.240336	0.236344	0.246761
Scale:	10062237.6	0.217982	0.214664	0.220526
Add:	8985643.0	0.361473	0.360575	0.363363
Triad:	8871850.0	0.366032	0.365200	0.366646



Human readable numbers:

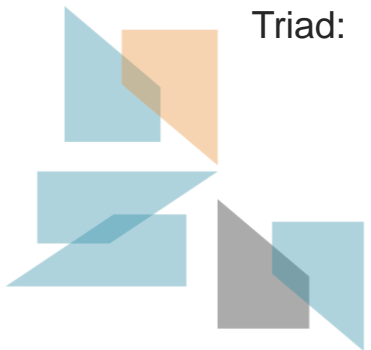
Array size = 3 TB

Copy = 9.1 TB/s

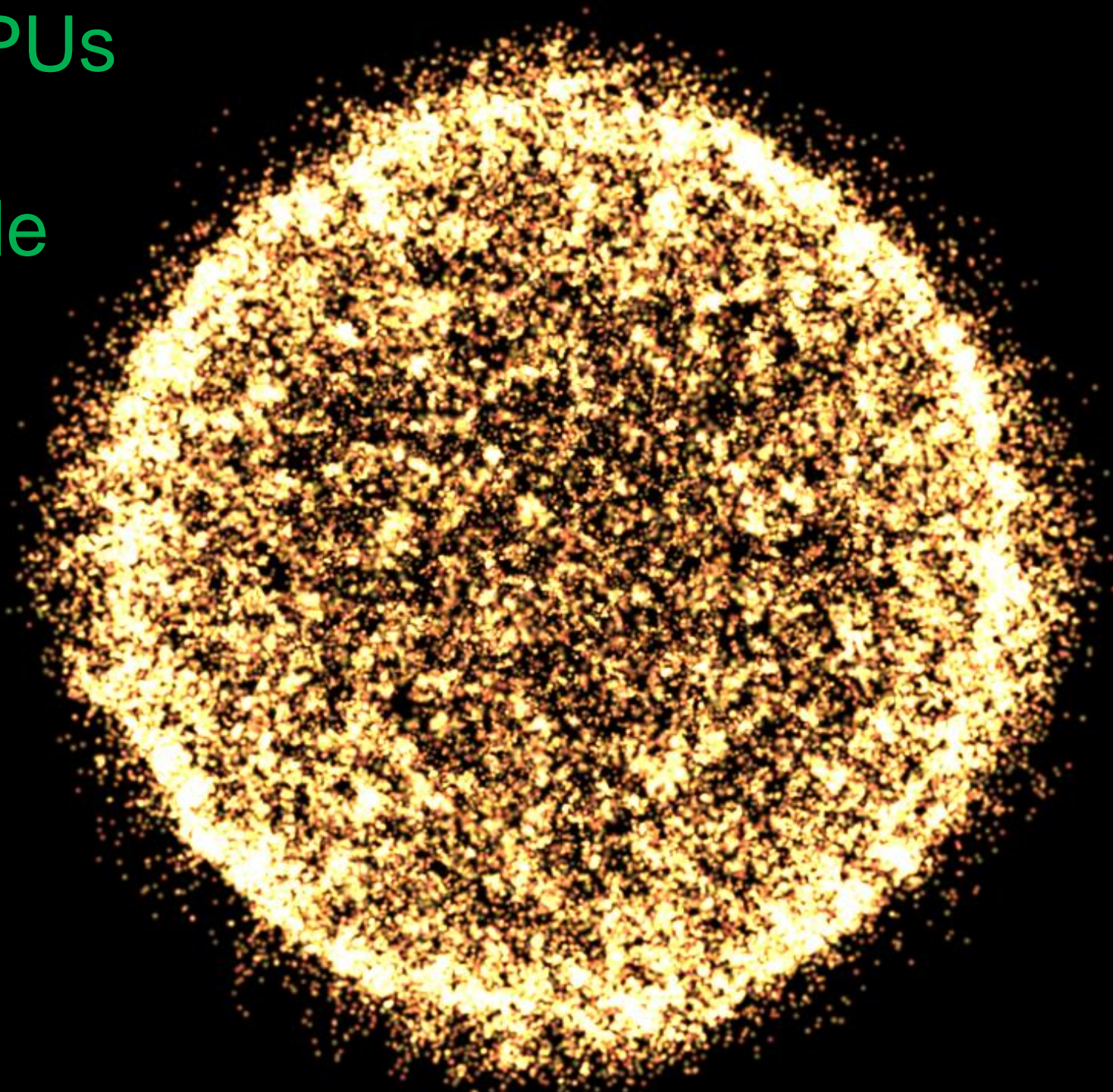
Scale = 10.0 TB/s

Add = 9.0 TB/s

Triad = 8.9 TB/s

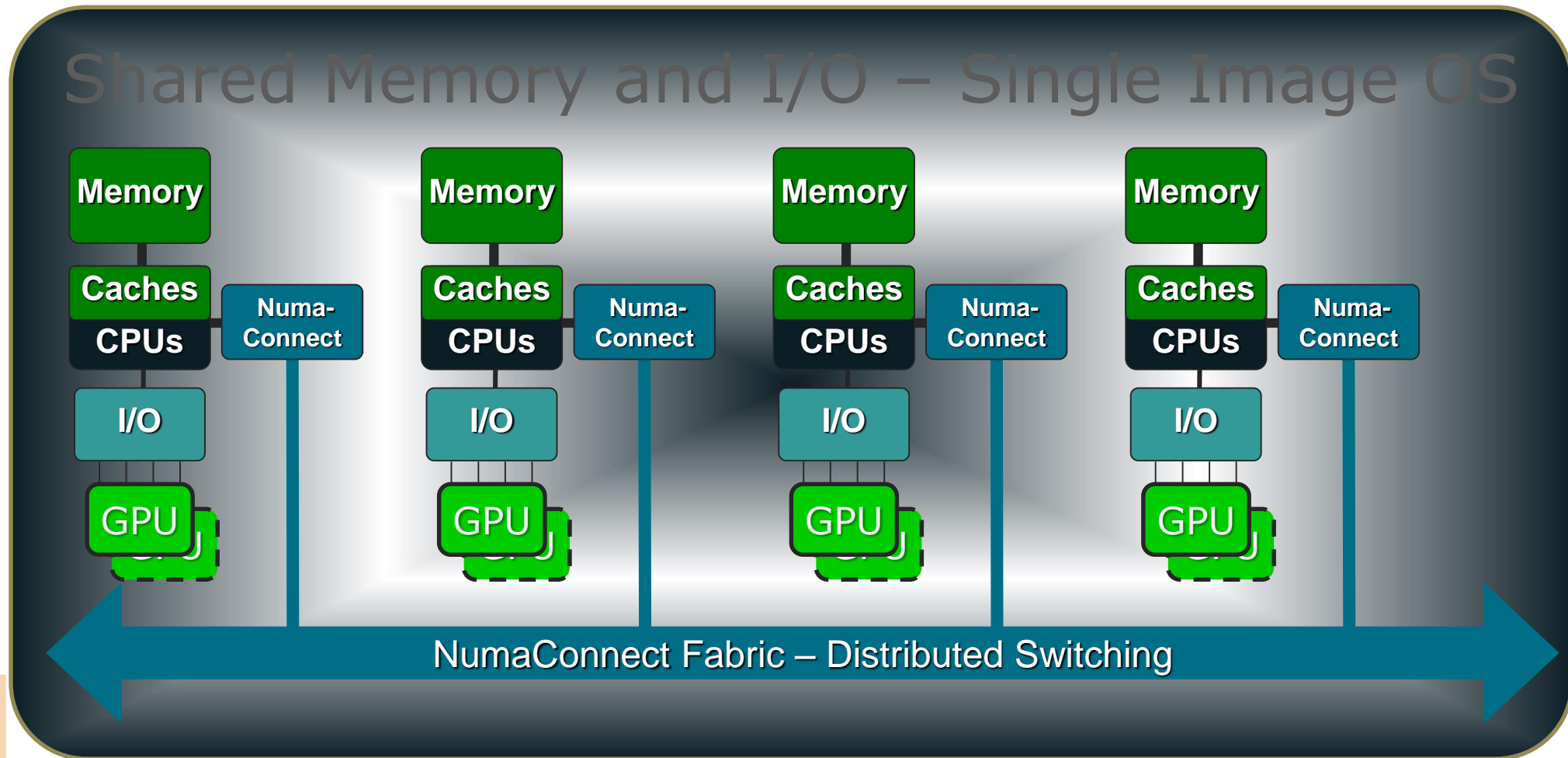


# Scalable GPUs with Numascale



# System Architecture

## Shared Memory and I/O – Single Image OS



# GPU Computing

- Standard GPU Clusters suffer from
  - Complex message based inter-node communication
  - Limited node size
  - Each node requires one instance of the OS
    - Complicates system administration
    - Complicates aggregation of GPUs
  - Underutilized CPUs



# Visualization in Clouds

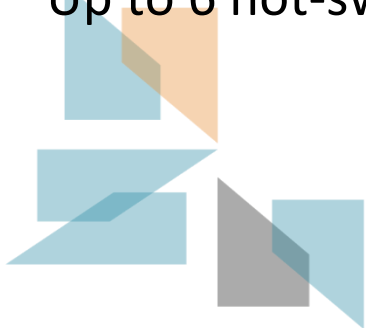
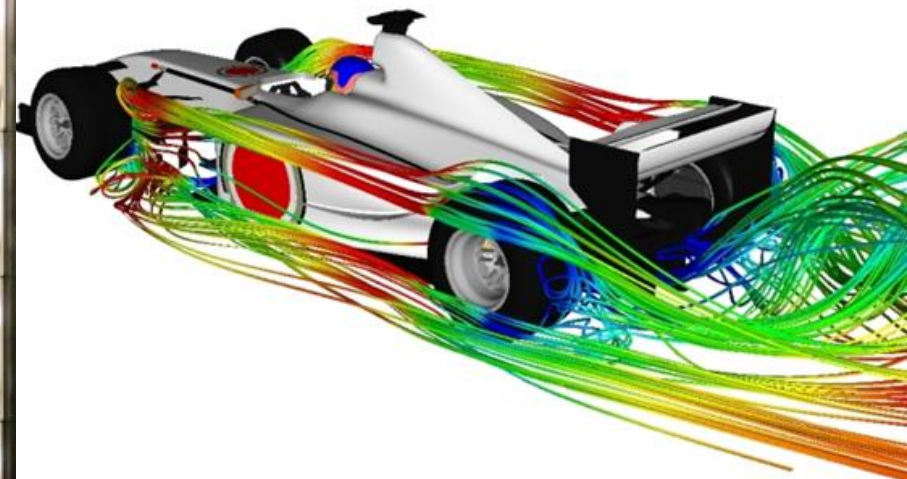
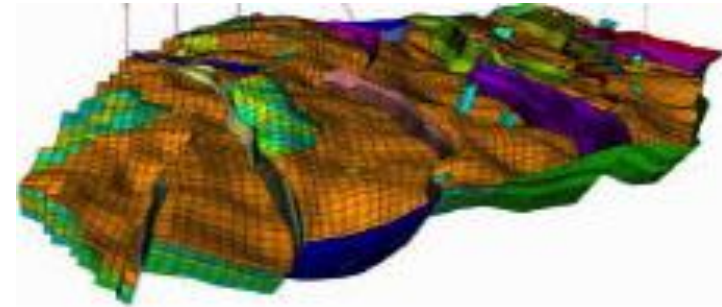
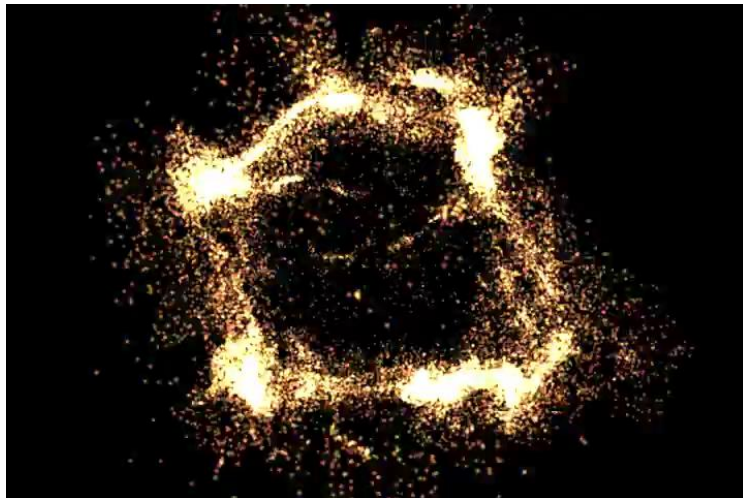
- Sharing GPU Resources
  - Fewer and larger systems
  - Better Resource Utilization
  - Less system administration
  - Easier upgrades



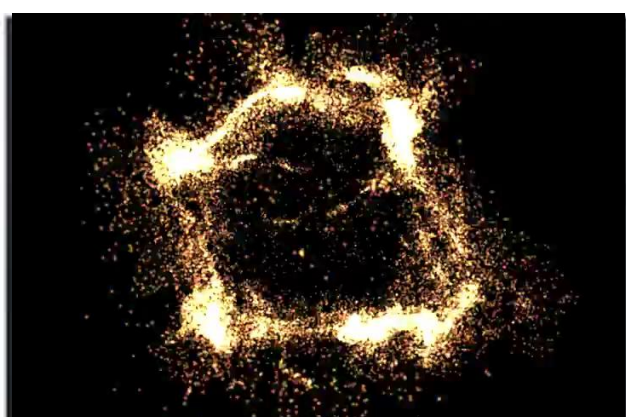
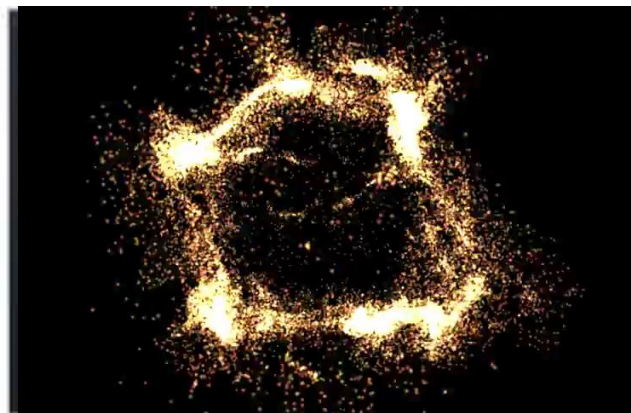
# Scalable GPUs

## Supermicro 2042 Server Features:

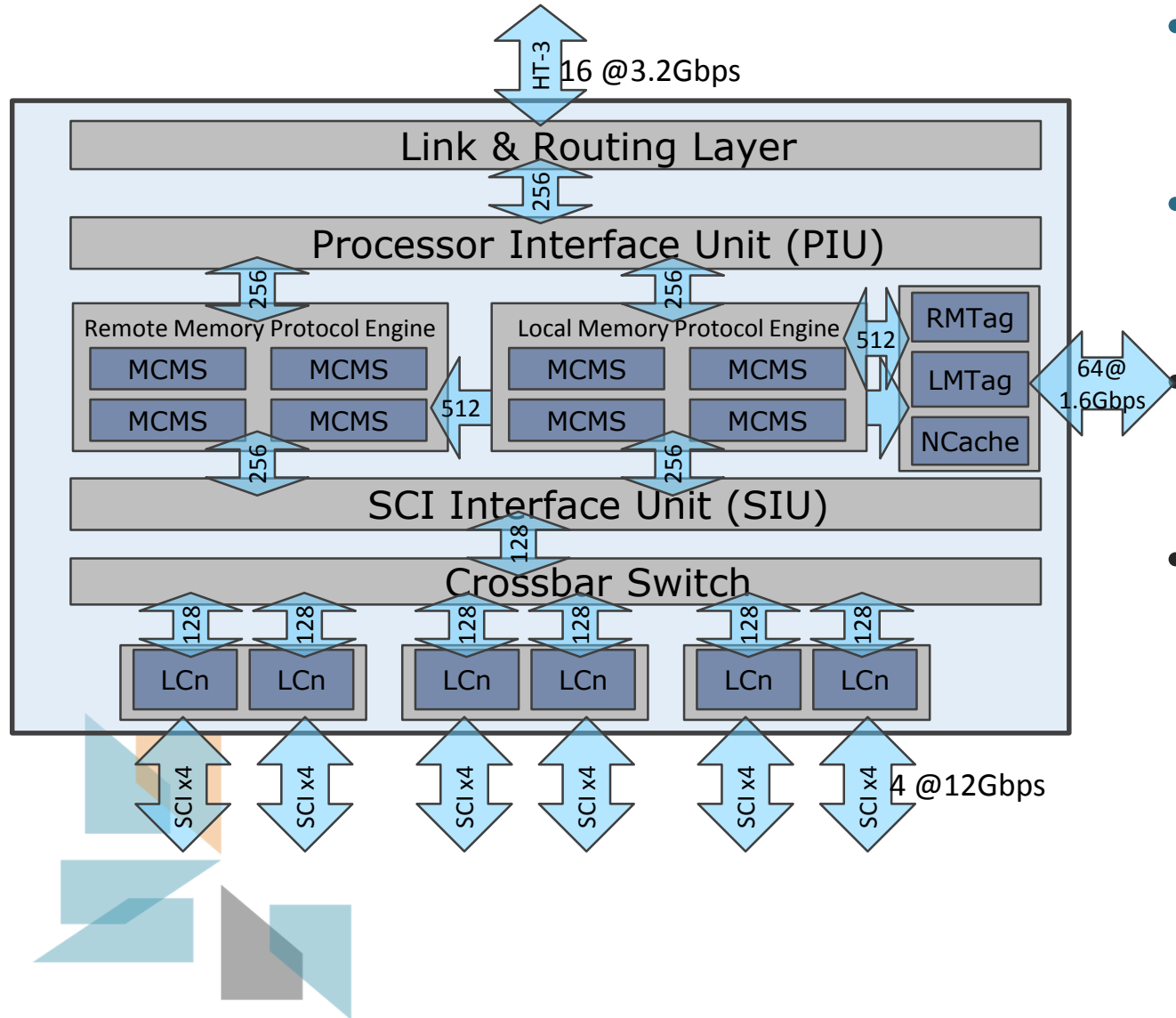
- Supermicro 2042G-LTF Server
- 2U Rack Mount
- 1 GPU - Single/Double Slot
- 2-3 AMD 63xx CPU
- 8 to 48 cores
- Up to 3.5GHz
- Up to 384 GB DRAM
- **NumaConnect Adapter**
- 1400W Redundant High-efficiency (Gold) Power Supplies
- Up to 6 hot-swap SAS/SATA



# 4 GPUs single image

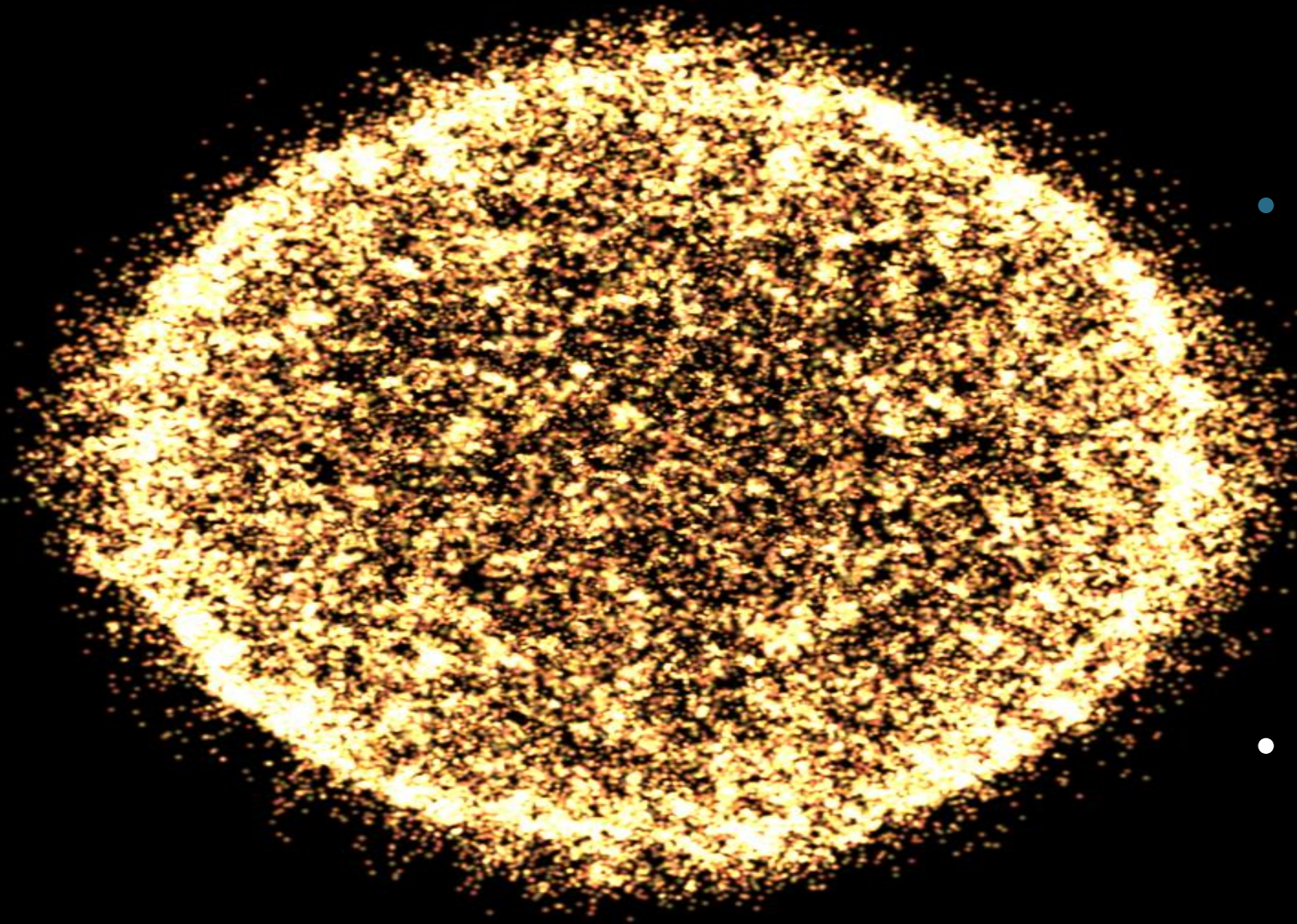
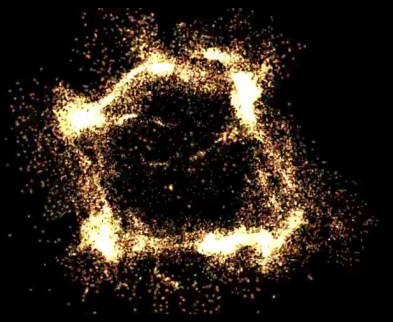


# News



- **NumaChip2** released with fatter pipes which gives more parallelism and BTE engine, both gives **more bandwidth**
- **Numascale R Appliance** released with optimized software stack and plug and play experience for Data Scientists
- Numascale has developed their own simple Jacobi solver that shows linear scaling for the Shyne code.
- Numascale is developing their own solver, for **solving** large **sparse** symmetric and unsymmetrical linear systems of equations on large shared-memory multiprocessors by maximum utilization of **NumaChip2**.

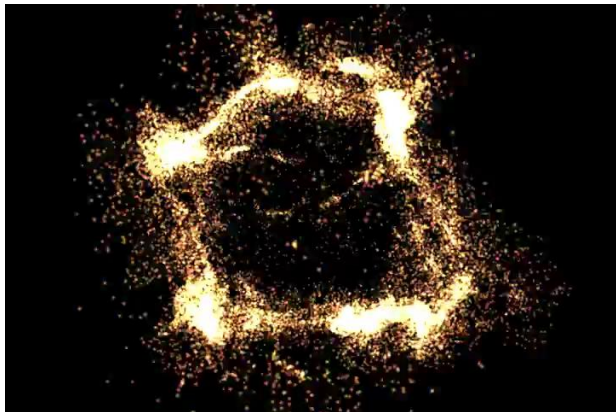
# Future system at Keele



- **One Numascale system**
  - 2,3 TB of memory
  - 576 cores
  - 12 GPUs.
- All resources are available in one Linux instance

24.36 FPS [single precision | 65536 bodies]  
2092.64 GFLOP/s on 3 GPUs

# Share everything



Thank you

Atle Vesterkjær  
Senior Software Engineer  
av@numascale.com  
February 2016

