The Tianhe's approach to IO-500: Experience & Practice

Russian Supercomputing Days, September 21-22, 2020

Prof. Ruibo Wang

National University of Defense Technology, China

CONTENTS

Part 1

About IO-500

Founders

Initial Steering Committee (aka. co-founders)

John Bent (Seagate)

Julian Kunkel (University of Reading)

Jay Lofstead (Sandia National Laboratories)

Other members (aka. Co-organizers)

Andreas Dilger (Whamcloud/DDN)

George Markomanolis (Oak Ridge National Laboratory)

“The steering committee is the decision body ensuring the development and curation of the benchmark and its results but also responsible to resolve ethical issues. “ - vi4io.org

History of the IO-500

Dec. 2015
High-performance Storage list

Jun. 2016
Lofstead and Kunkel's ISC talk

Nov. 2016
joint BoF from Kunkel, Bent and Lofstead at SC

Jun. 2017
approach, benchmarks, rules released at ISC & Selected runs on Top500 systems

Nov. 2017
1st result shown during SC

Nov. 2018
Top500 #1 system Summit participated and won.

Nov. 2019
WekaIO-Matrix won the #1 in the ranked list. Tianhe-2E got the best BW award

Jul. 2020
Intel DAOS won the #1 and the score is twice of the 2nd

IO-500 benchmark

Lists

(1) Full list - includes all submissions. Systems may be deployed at arbitrary scales.
(2) Historic list - include all submissions in the history.
(3) 10-node list - Submissions use only 10 client nodes for the benchmark. The number of servers is not limited.

What to test

IOR Easy - Measures the most efficient I/O pattern, users can declare the parameters and save 1 file per process.
IOR Hard - Single-shared file, 47008 byte random access
MD Easy - Create rank directories with N empty files.
MD Hard - Single-shared directory, files of 3901 bytes
Find - Searches for files of 3901 bytes across all created files.

How's the score calculated/released?

IO500

$$Score_{meta} = \sqrt[8]{\left(\prod_{1}^{8} IOPS_i\right)}$$

$$Score_{bw} = \sqrt[4]{\left(\prod_{1}^{4} BW_i\right)}$$

$$Score = \sqrt{Score_{meta} * Score_{bw}}$$

- The scores for submissions release at every SC/ISC conference (aka "around every June and November").

- So essentially geometric mean is used – honors tuning equally but insensitive to 'outliers'. That means IO-500 encourages balanced systems.

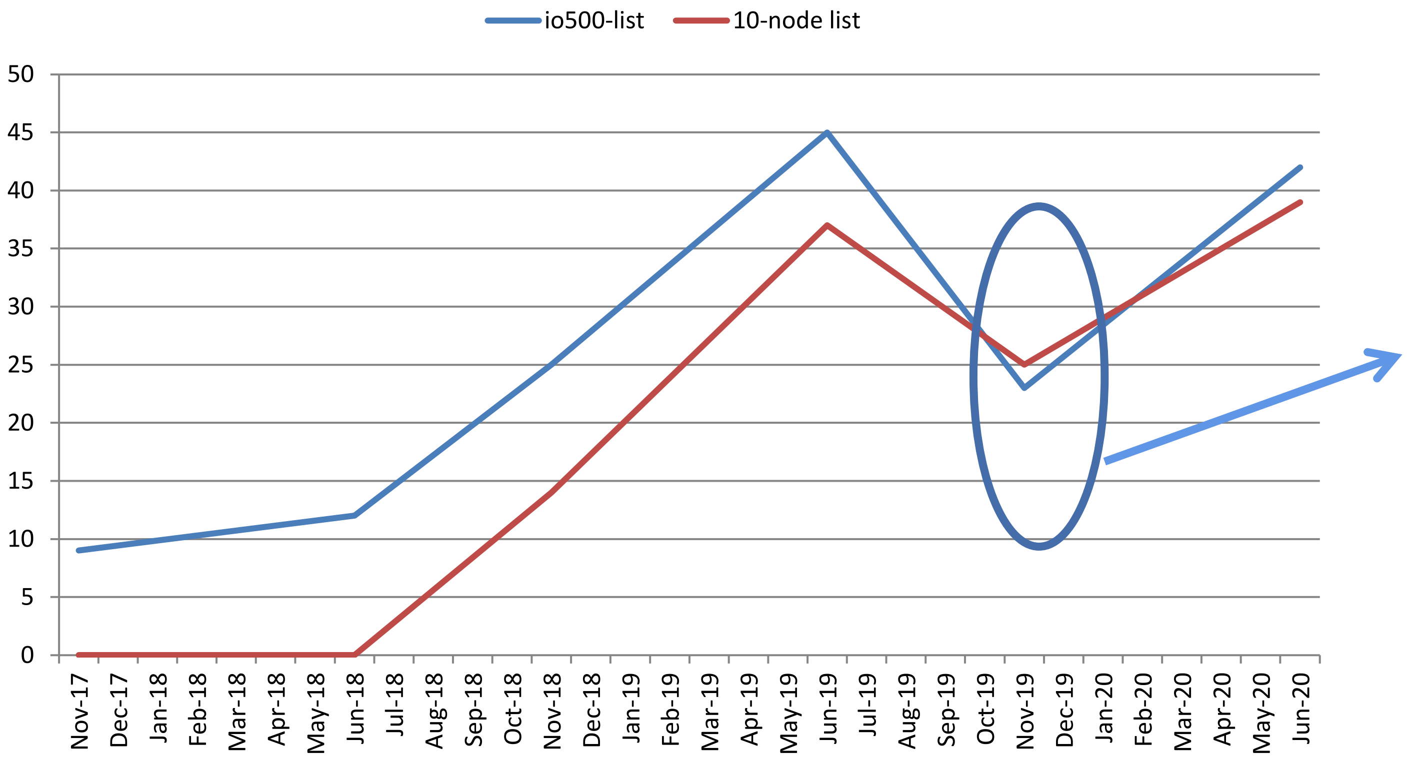
-The latest IO-500 runs both script/C version of the benchmark as a cross validation. Future tests will only use the C version.

PART 2

The current trend

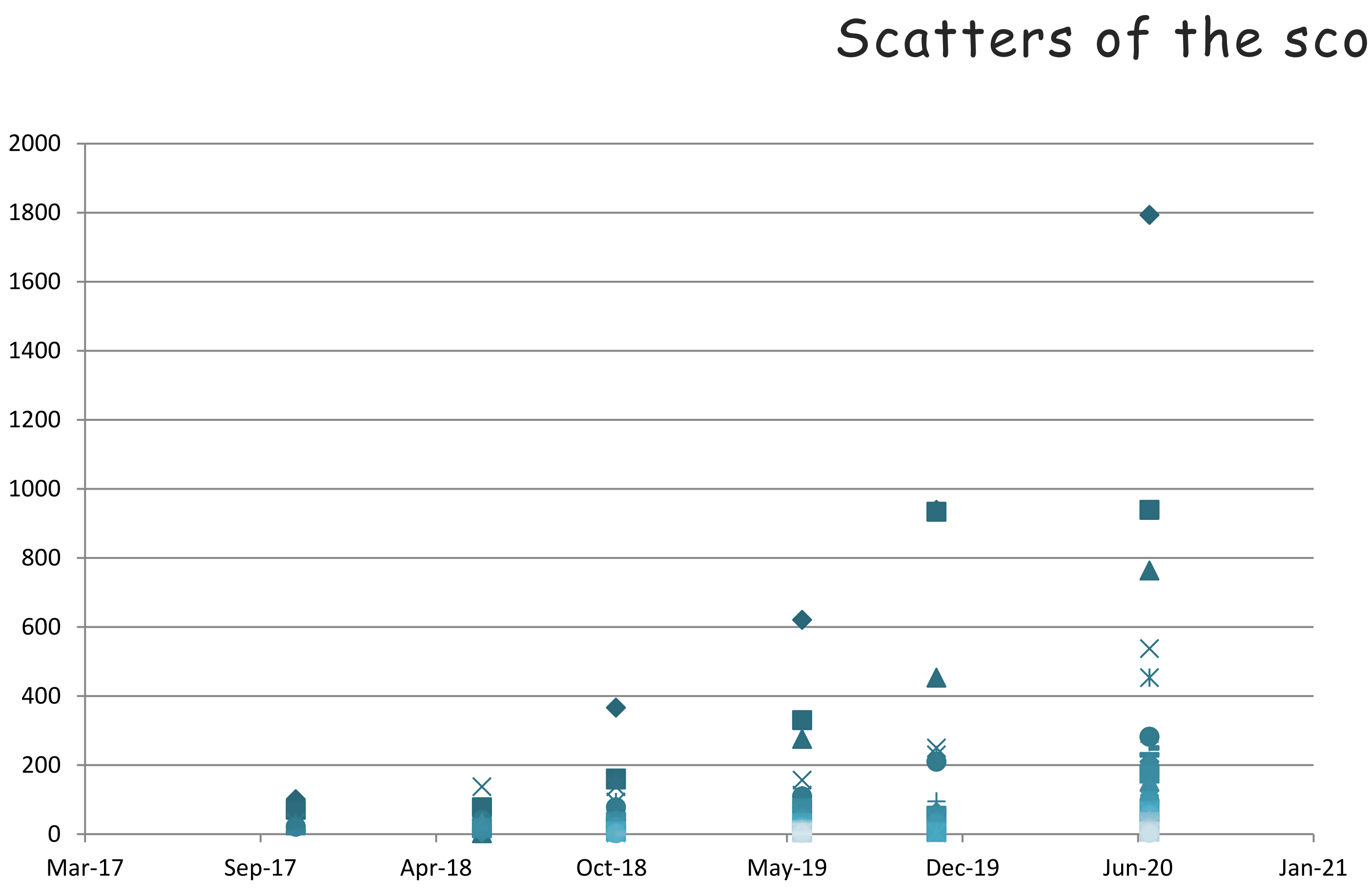
From past to now

The number of tested systems



Rule changed – mdtest adds shift reader ranks

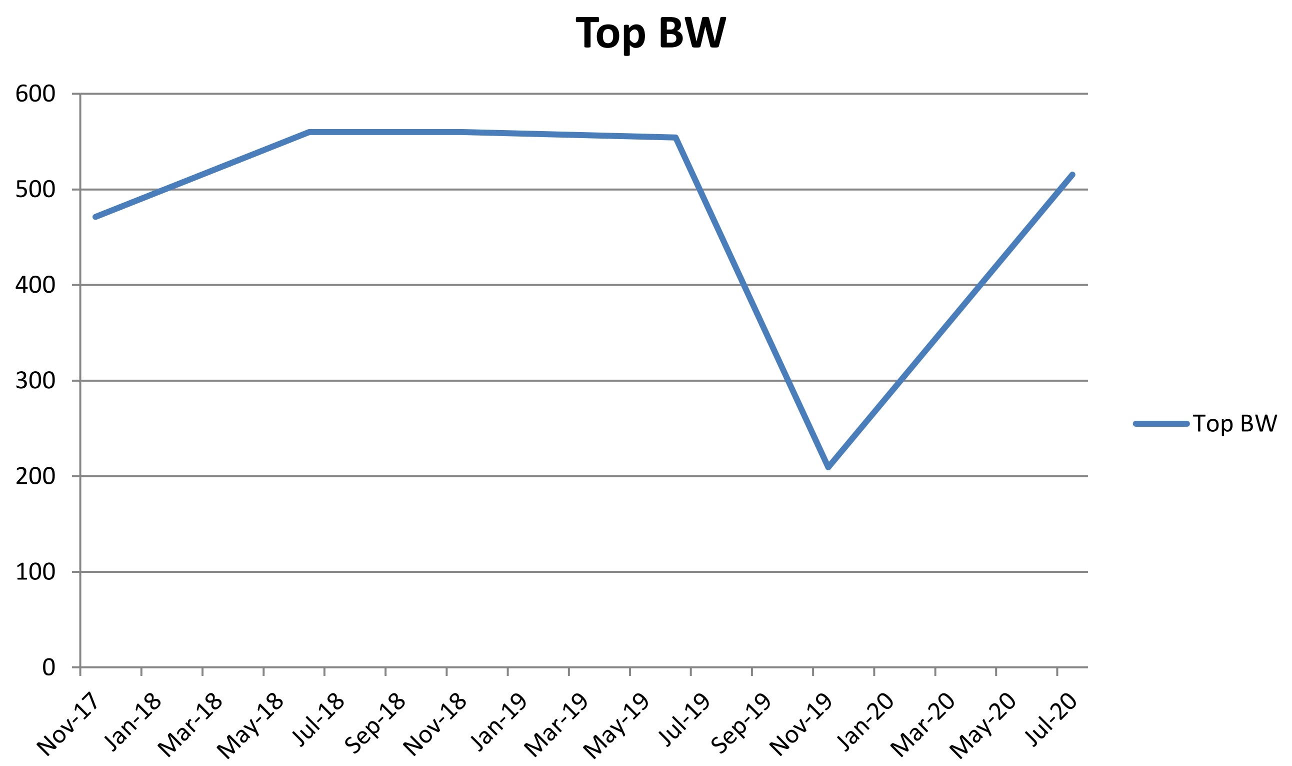
From past to now



The performance gap between the top system and the rest are becoming larger.

From past to now

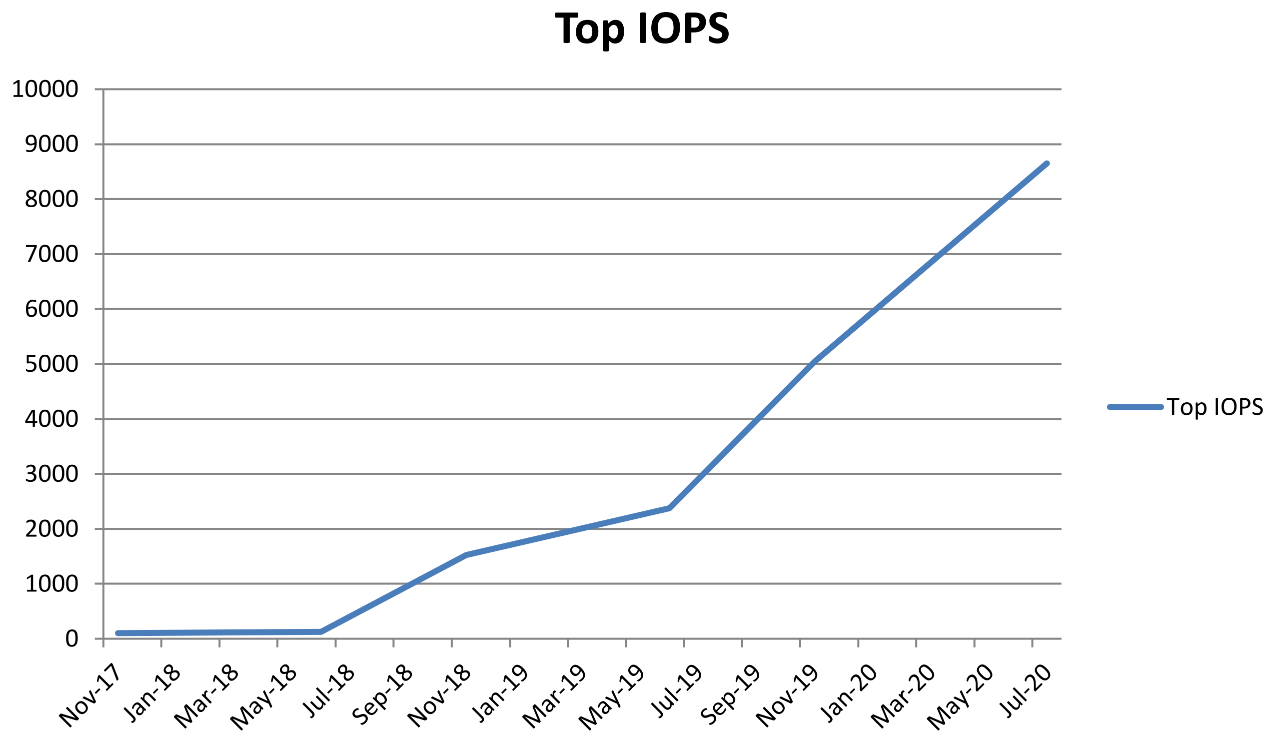
TOP bandwidth trend



The TOP bandwidth is almost stable over years

From past to now

TOP IOPS trend



The TOP IOPS is growing almost linearly. Latest top systems win by IOPS rather than bandwidth.

Winning Techniques

01

BW & Burst Buffers

Burst buffer systems like DDN IME are pushing forward the limit of bandwidth. Such systems achieved BW of ~500+GB/s in the IO500 benchmarks.

02

Metadata & New Storage

New storage designs, such as DAOS and WekaIO Matrix, avoid the expensive overhead of OS context switch, thus boosting the metadata performance.

TOP500 Systems in IO-500

So far, very few systems in the TOP-500 list rank well in the IO-500 ranking list. **Summit from ORNL** (No. 1 on the TOP 500 - Nov.2018) used to be the No.1 in Nov. 2018 but surpassed by a Lustre-based system from U of Cambridge in June, 2019.

Summit's Hardware & Spectrum Scale (formerly GPFS) file system :

- 77 GL4 Elastic Storage Server (network: dual-port Mellanox IB EDR)
- Capacity 250PB, Bandwidth 2.5TB/s
- Burst Buffer (Local NVMe SSDs) Capacity 7.4PB, Bandwidth 9.7GB/s

Their IO-500 testing system:

- 1008 client procs (2 on each of the 504 client nodes)
- 154 md/ds nodes (each storage server is also a metadata server)

TOP500 Systems in IO-500

Metric	ior_easy_w	ior_easy_r	ior_hard_w	ior_hard_r	md_easy_w	md_easy_stat
Perf. (GB/s & kIOPS)	2158.7	1788.32	0.57	27.4	3071	28769.4

Metric	md_easy_del	md_hard_w	md_hard_r	md_hard_stat	md_hard_del	find_hard
Perf. (kIOPS)	2699	24.38	15354.7	560.89	26.5	21780.03

Summit got a total score of 366.47
- 1.Increasing number of segments;
- 2.Decreasing the number of processes per node.

TOP500 Systems in IO-500

Fugaku is currently the fastest supercomputer in the world.

Fugaku and Oracle are collaborating to enable cloud-based storage option for HPC. Their testing system spec:

- BeeGFS BeeOND
- 270 nodes at Oracle as storage servers, 170 client nodes with 2040 procs.

Although achieving the performance of Linpack 415.5 Petaflops, Fugaku only got **the 7th place** in the IO-500 list in this year

TOP500 Systems in IO-500

Metric	ior_easy_w	ior_easy_r	ior_hard_w	ior_hard_r	md_easy_w	md_easy_stat
Perf. (GB/s & kIOPS)	333.85	422.2	35.72	187.34	1913.3	8261.12

Metric	md_easy_del	md_hard_w	md_hard_r	md_hard_stat	md_hard_del	find_hard
Perf. (kIOPS)	3602.59	2.42	18.24	25.9	8.34	6537.04

Fugaku suffers from low metadata performance.

Storage matters a lot for HPC

- Amdahl says any computer system's performance is limited by its slowest component.
- As the HPC is going towards 100 million threads, we demand new approach to I/O for overcoming the **Exascale challenge**.
- Reading/writing data to parallel file systems are major bottlenecks.
- We may need to improve the inter-processor communication and rethink the design of storage systems.

Technical trend for winning systems

Winning systems trend (TOP5)

2017-11	2018-06	2018-11	2019-06	2019-11	2020-07
IME (JCAHPC)	IME (JCAHPC)	*Spectrum Scale(Oak Ridge)	Lustre (U of Cambridge)	WekaIO (WekaIO)	DAOS (Intel)
DataWarp (KAUST)	DataWarp (KAUST)	IME (KISTI)	*Spectrum Scale(Oak Ridge)	DAOS (Intel)	WekaIO (WekaIO)
Lustre (KAUST)	Lustre (KAUST)	Lustre (U of Cambridge)	IME (JCAHPC)	*Lustre (NUDT)	DAOS (TACC)
BeeGFS (JSC)	BeeGFS (JSC)	IME (JCAHPC)	IME (KISTI)	Lustre (NVIDIA)	DAOS (Argonne)
Lustre (DKRZ)	Lustre (DKRZ)	WekaIO (WekaIO)	IME (DDN)	Lustre (U of Cambridge)	*Lustre (NUDT)

Burst buffer systems, Lustre, GPFS -> WekaIO Matrix and Intel DAOS

Emerging systems keep breaking the records

Almost all top systems are DAOS (from the historic data until ISC20). WekaIO Matrix also performs non-trivially.

#	information								io500		
	list id	institution	system	storage vendor	filesystem type	client nodes	client total procs	data	score	bw	md
										GiB/s	kIOP/s
1	isc20	Intel	Wolf	Intel	DAOS	52	1664	zip	1792.98	371.67	8649.57
2	isc20	Intel	Wolf	Intel	DAOS	52	1664	zip	1614.02	368.44	7070.51
3	sc19	WekaIO	WekaIO on AWS	WekaIO	WekaIO Matrix	345	8625	zip	938.95	174.74	5045.33
4	sc19	Intel	Wolf	Intel	DAOS	26	728	zip	933.64	183.36	4753.79
5	isc20	TACC	Frontera	Intel	DAOS	60	1440	zip	763.80	78.31	7449.56
6	isc20	Intel	Wolf	Intel	DAOS	10	420	zip	758.71	164.77	3493.56
7	isc20	TACC	Frontera	Intel	DAOS	60	1440	zip	756.01	75.40	7580.23
8	isc20	Intel	Wolf	Intel	DAOS	10	420	zip	736.33	163.19	3322.40

Behind the curtain

It's natural to ask why these systems are doing much better?

The devices are changing but POSIX does not fit.

Traditional parallel file systems with POSIX standard are designed in the era of spinning disks to hide the latency of devices that could only manage a few hundreds IOPS.

Behind the curtain

- POSIX-based parallel file systems provide strong consistency semantics but modern HPC applications do not need. DAOS is designed to bypass the POSIX I/O bottlenecks.

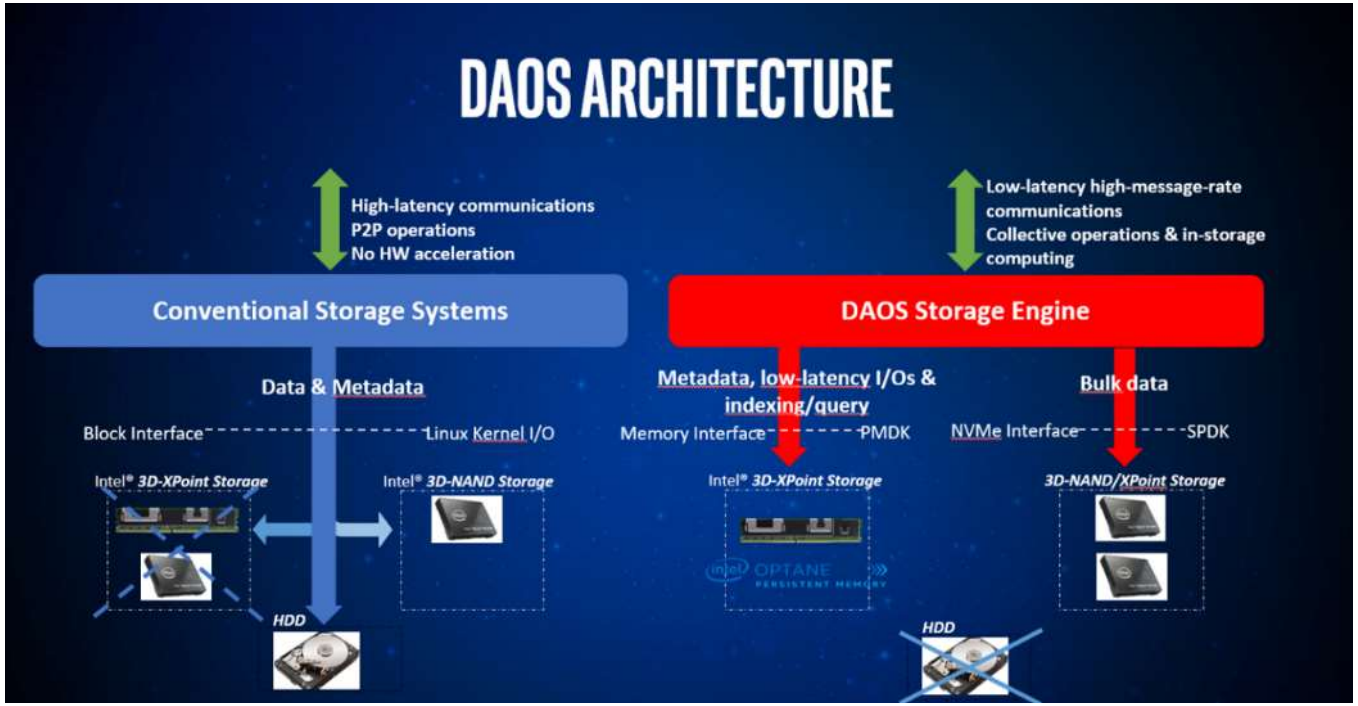


Fig from https://daos-stack.github.io/overview/architecture/

Behind the curtain

Rationales of relaxing POSIX?

(1) For very large scales, relying on the file system to provide consistency is quite inefficient. The applications could just ensure no two processes are trying to write to the same part of a file at once.
(2) In multi-tenants systems, it is extremely rare for two independent jobs to manipulate the same file. – e.g., DAOS provides containers as independent object address spaces.

Consistency guarantee like POSIX is overkill in many cases!

Behind the curtain

DAOS relaxes the POSIX consistency in the following ways:

(1) A baseline model – individual I/O operations are tagged with different epoch and applied in such order. (delivers the max scalability and performance for applications that do not generate conflicting I/Os)
(2) A distributed serializable transaction based on multi-version concurrency control. (sort of lockless optimistic concurrency control for applications that require conflict serialization)

Behind the curtain

DAOS uses **NVDIMM** to store the **metadata and small writes** in a key-value manner. Only large **bulk data** to **SSDs**.

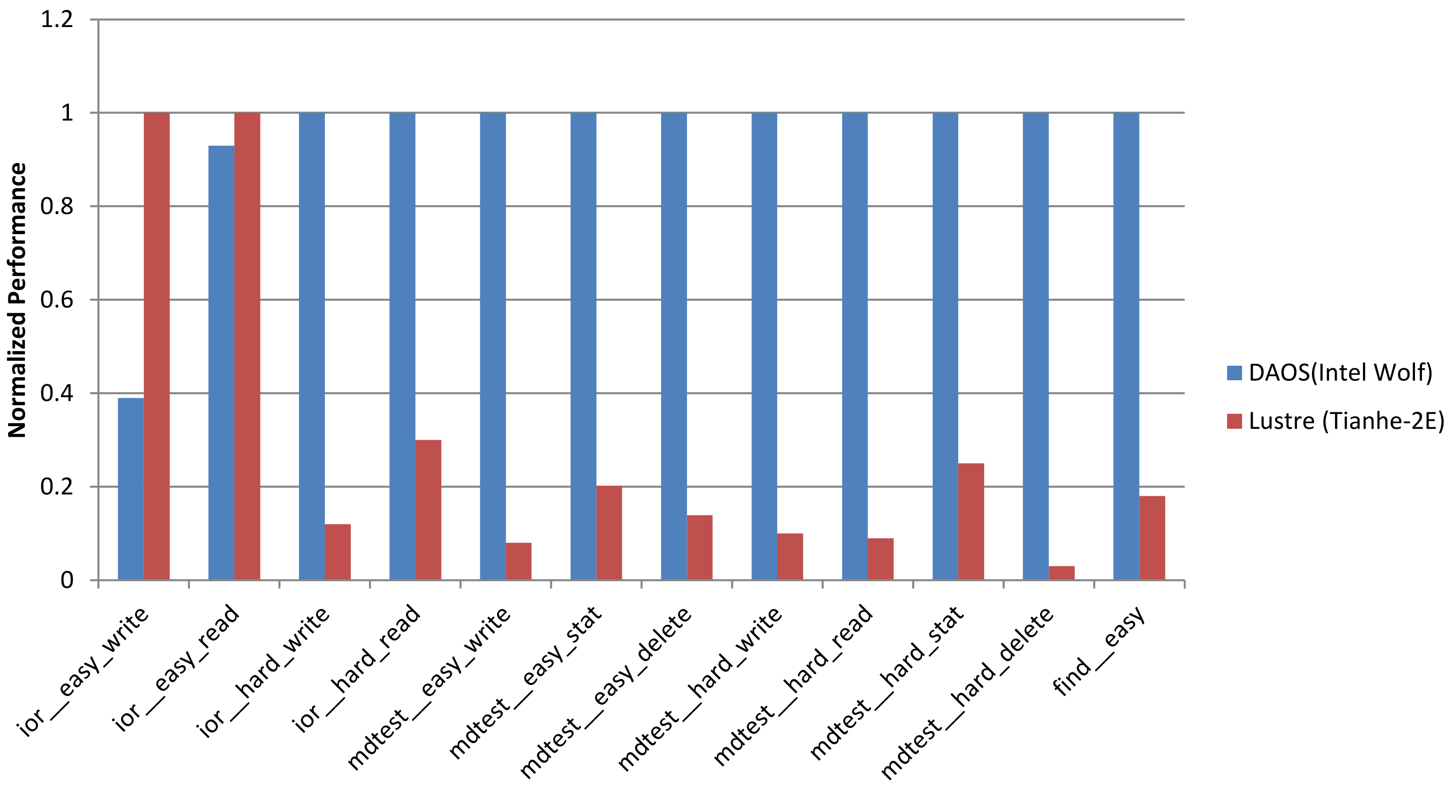
Full user-space model without system calls on the I/O path. Avoids the OS overhead.

Fig from *https://www.alcf.anl.gov/sites/default/files/2020-03/2020-03%20slide%20presentation%20DAOS.pdf

Behind the curtain

Comparing DAOS with Lustre (the performance of top systems)

Intel Wolf v.s. NUDT Tianhe-2E



DAOS wins mostly by high IOPS (a result of having less indirections on the storage stack)

PART 3

Our practice

Tianhe-2E

deployed in the National Supercomputer Center in Changsha, 2019

500 compute nodes

72 storage servers

Lustre 2.12.2 file system

Compute Racks

Storage Racks

Tianhe-2E

NUDT's Tianhe-2E is so far the highest ranked system in China.

Nov. 2019, rank #3, BW #1
Jul. 2020, rank #5

Today, Tianhe-2E is well-optimized as it is the **only top system** which **uses Lustre** file system rather than emerging object-based systems.

1	sc19	WekaIO	WekaIO on AWS	WekaIO	WekaIO Matrix
2	sc19	Intel	Wolf	Intel	DAOS
3	sc19	National Supercomputing Center in Changsha	Tianhe-2E	National University of Defense Technology	Lustre
4	sc19	NVIDIA	DGX-2H SuperPOD	DDN	Lustre
5	sc19	University of Cambridge	Data Accelerator	Dell EMC	Lustre

Tianhe-2E in IO-500

Metric	ior_easy_w	ior_easy_r	ior_hard_w	ior_hard_r	md_easy_w	md_easy_stat
Perf. (GB/s & kIOPS)	608.64	613.53	31.87	161.64	1099.18	2416.4

Metric	md_easy_del	md_hard_w	md_hard_r	md_hard_stat	md_hard_del	find_hard
Perf. (kIOPS)	961.62	554.07	779.21	2419.15	203.5	1603.06

Tianhe-2E got the bandwidth award in Nov. 2019.

SC 2019

2019-11

This is the official list from Supercomputing 2019. The list shows the best result for a given combination of system/institution/filesystem.

Please see also the 10 node challenge ranked list.

#	information								io500		
	list id	institution	system	storage vendor	filesystem type	client nodes	client total procs	data	score	bw GiB/s	md kIOP/s
1	sc19	WekaIO	WekaIO on AWS	WekaIO	WekaIO Matrix	345	8625	zip	938.95	174.74	5045.33
2	sc19	Intel	Wolf	Intel	DAOS	26	728	zip	933.64	183.36	4753.79
3	sc19	National Supercomputing Center in Changsha	Tianhe-2E	National University of Defense Technology	Lustre	480	5280	zip	453.68	209.43	982.78
4	sc19	NVIDIA	DGX-2H SuperPOD	DDN	Lustre	10	400	zip	249.50	86.97	715.76
5	sc19	University of Cambridge	Data Accelerator	Dell EMC	Lustre	128	2048	zip	229.45	131.25	401.13

SC 2019

Test spec – 52 storage servers (dual-OPA, 12 NVMe SSDs, 40 metadata servers), 480 client nodes (11 processes on each node)

With proper optimizations and tunings, such as

1. dual-OPA card configuration.
2. tuning the IO segment size.
3. NUMA affinity configurations.

We got the IO-500 BW award.

ISC 2020

Using the same storage servers, we prepared our IO-500 runs with Intel DAOS. We did not submit the result because of the network performance issues.

```
[root@mn0%yhstar ok]# cat 2020.07.13-23.52.13-app/result_summary.txt
IO500 version io500-isc20_v2-5-g34cd5fde4ce7-39
[RESULT]       ior-easy-write      211.413142 GiB/s  : time 327.034 seconds
[RESULT]    mdtest-easy-write     4281.981779 kIOPS : time 313.915 seconds
[RESULT]       ior-hard-write       98.113964 GiB/s  : time 304.415 seconds
[RESULT]    mdtest-hard-write     1241.133440 kIOPS : time 307.602 seconds
[RESULT]                 find     1693.736193 kIOPS : time 327.850 seconds
[RESULT]        ior-easy-read      135.068322 GiB/s  : time 509.562 seconds
[RESULT]     mdtest-easy-stat     5642.004805 kIOPS : time 239.664 seconds
[RESULT]        ior-hard-read       50.244430 GiB/s  : time 590.392 seconds
[RESULT]     mdtest-hard-stat     4979.051173 kIOPS : time 79.943 seconds
[RESULT]   mdtest-easy-delete     2638.980205 kIOPS : time 514.213 seconds
[RESULT]     mdtest-hard-read     1155.849036 kIOPS : time 330.105 seconds
[RESULT]   mdtest-hard-delete     2449.846975 kIOPS : time 343.741 seconds
[SCORE] Bandwidth 108.924621 GB/s : IOPS 2567.708173 kiops : TOTAL 528.854081
```

ISC 2020

We also benchmarked the 10-node performance.

```
[root@mn0%yhstar ok]# cat 2020.07.09-00.15.49-app/result_summary.txt
IO500 version io500-isc20_v2-5-g34cd5fde4ce7-39
[RESULT]       ior-easy-write      168.239467 GiB/s  : time 310.950 seconds
[RESULT]    mdtest-easy-write     2508.667508 kIOPS : time 304.177 seconds
[RESULT]       ior-hard-write       50.135789 GiB/s  : time 305.560 seconds
[RESULT]    mdtest-hard-write      928.602420 kIOPS : time 312.045 seconds
[RESULT]                 find     1053.524558 kIOPS : time 318.774 seconds
[RESULT]        ior-easy-read      114.875313 GiB/s  : time 451.417 seconds
[RESULT]     mdtest-easy-stat     2858.373017 kIOPS : time 266.231 seconds
[RESULT]        ior-hard-read       22.072284 GiB/s  : time 688.842 seconds
[RESULT]     mdtest-hard-stat     2533.901754 kIOPS : time 116.363 seconds
[RESULT]   mdtest-easy-delete     1494.132445 kIOPS : time 513.426 seconds
[RESULT]     mdtest-hard-read     1059.409222 kIOPS : time 275.002 seconds
[RESULT]   mdtest-hard-delete     1383.912682 kIOPS : time 342.162 seconds
[SCORE] Bandwidth 68.004465 GB/s : IOPS 1580.514461 kiops : TOTAL 327.844537
```

2-3 Lessons learned

01

OPA networks are not quite stable, we spent much time to get psm2 drivers to work properly.

```
diff --git a/ptl_ips/ips_proto_connect.c b/ptl_ips/ips_proto_connect.c
index a608760..f395f8d 100644
--- a/ptl_ips/ips_proto_connect.c
+++ b/ptl_ips/ips_proto_connect.c
@@ -1537,10 +1537,10 @@ ips_proto_disconnect(struct ips_proto *proto, int force, int numep,
                        /* Remote disconnect req arrived already, remove this epid.  If it
                         * hasn't arrived yet, that's okay, we'll pick it up later and just
                         * mark our connect-to status as being "none". */
-                       if (ipsaddr->cstate_incoming == CSTATE_NONE) {
+                       // if (ipsaddr->cstate_incoming == CSTATE_NONE) {
                                ips_free_epaddr(array_of_epaddr[i], proto);
                                array_of_epaddr[i] = NULL;
-                       } else
+                       // } else
                                ipsaddr->cstate_outgoing = CSTATE_NONE;
                }
```

2-3 Lessons learned

02

DER_TIMEOUT error while creating pools/containers.

This is mainly due to the fact of using tmpfs to support NVRAM. One needs to set PMEMOBJ_CONF=prefault.at_create = 1 which forces each page to be allocated. Also, we need to set the CRT_TIMEOUT to a larger value (~10min) as the creation can sometimes take quite a while.

03

Config at the client side.

```
export FI_PSM2_DISCONNECT=1
export CRT_CTX_SHARE_ADDR=0
```

2-3 Lessons learned

04 Enabling dual-OPA configuration to boost the performance.

```
diff --git a/src/client/api/init.c b/src/client/api/init.c
index 9d1f1ce..82eefc9 100644
--- a/src/client/api/init.c
+++ b/src/client/api/init.c
@@ -40,6 +40,8 @@
 #include <daos/placement.h>
 #include "task_internal.h"
 #include <pthread.h>
+#include <unistd.h>
+#include <numa.h>

 static pthread_mutex_t module_lock = PTHREAD_MUTEX_INITIALIZER;
 static bool            module_initialized;
@@ -143,6 +145,24 @@ int
 daos_init(void)
 {
     int rc;
+    char    *iface;
+
+    iface = getenv("OFI_INTERFACE");
+    if (iface == NULL) {
+        int     cpu;
+        char    *hfi = NULL;
+
+        cpu = sched_getcpu();
+        if (numa_node_of_cpu(cpu) == 0) {
+            iface = "ib0";
+            hfi = "0";
+        } else {
+            iface = "ib1";
+            hfi = "1";
+        }
+        setenv("OFI_INTERFACE", iface, 1);
+        setenv("HFI_UNIT", hfi, 1);
+    }

    D_MUTEX_LOCK(&module_lock);
    if (module_initialized)
```

2-3 Lessons learned

05

of CPU cores matters.

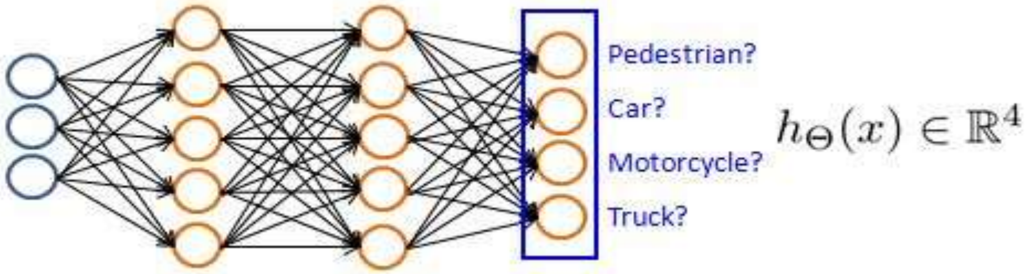
We use 24 servers (Xeon 8 cores, 2 sockets) and 24 clients (the same spec). The corresponding Intel runs normally have CPUs with 20+ cores. # of CPU cores especially limit # of client processes, thus hurting our 10-node performance. Also, that means we need more nodes (which we did not have by the time of submission) to act as clients in our full run.

PART 4

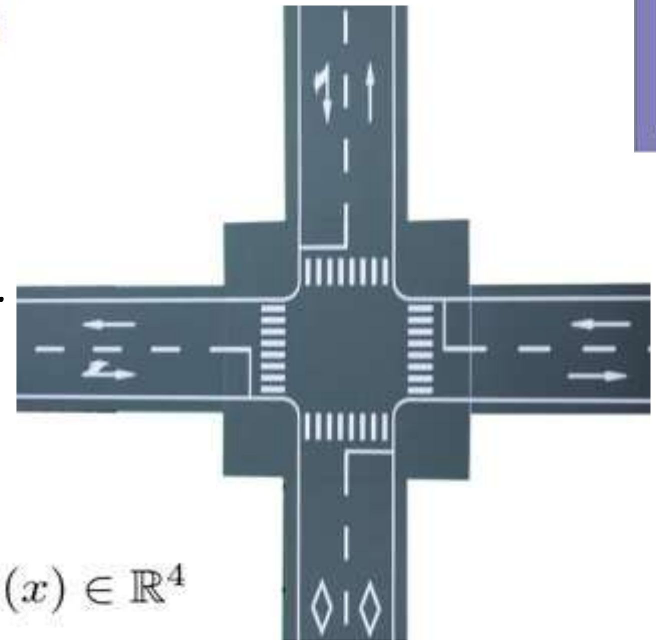
Conclusion

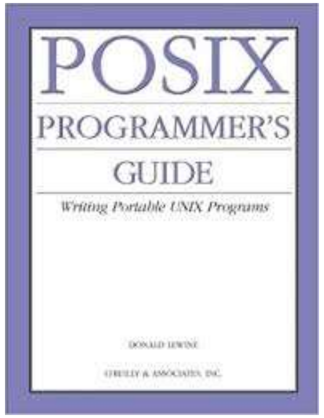
Challenges & Opportunities

New storage devices are emerging…



HPC applications are evolving…





POSIX may be outdated? But new standard is not mature yet.

Q & A

T H A N K S