MadFS高性能分布式缓存文件系统的设计与实现

陈康 王润基
清华大学
鹏城实验室

鹏城云脑II配置参数

- 512 Nodes
- CPU: Kunpeng 920 ARMv8
  4 sockets x 48 cores, 2.6GHz
- RAM: 2T, DDR4-2666
- NIC: HiSilicon 100GE x2
  with RoCE RDMA
- SSD: NVMe SSD 3.2T x6

IO500

- 计算方法：IO500包括带宽和元数据两项基准测试，将两个项目的总分（总分也是分项的几何平均）进行几何平均后，得出最终分数。
- IO：写入数据，再读出
- MD：创建、删除、写入文件
- Find: 查找
- 大的分下面五项内容
  - IOEasy：IO模式充分优化的应用程序
  - IOHard：需要进行随机读写的应用程序
  - MDEasy：元数据读写/小文件（小对象）的读写
  - MDHard：共享目录下的小文件（3901个字节）读写
  - Find：基于模式的对象查找

IO500 SC21 LIST

IO500 SC21 List

IO500 | 10 Node | Full | Historical | Customize | Download

This is the SC21 IO500 list

#	INFORMATION							IO500	
	BOF	INSTITUTION	SYSTEM	STORAGE VENDOR	FILE SYSTEM TYPE	CLIENT NODES	TOTAL CLIENT PROC.	SCORE	BW (GIB/S)	MD (KIOP/S)
1	ISC21	Pengcheng Laboratory	Pengcheng Cloudbrain-II on Atlas 900	Pengcheng	MadFS	512	36,864	36,850.40	3,421.62	396,872.82
2	SC21	Huawei HPDA Lab	Athena	Huawei	OceanFS	10	1,720	2,395.03	314.56	18,235.71
3	SC21	Olympus Lab	OceanStor Pacific	Huawei	OceanFS	10	1,720	2,298.69	317.07	16,664.88
4	SC21	Huawei Cloud		PDSL	Flashfs	15	1,560	2,016.70	109.82	37,034.00
5	ISC21	Intel	Endeavour	Intel	DAOS	10	1,440	1,859.56	398.77	8,671.65
6	ISC20	Intel	Wolf	Intel	DAOS	52	1,664	1,792.98	371.67	8,649.57
7	ISC21	Lenovo	Lenovo-Lenox	Lenovo	DAOS	36	3,456	988.99	176.37	5,545.61
8	SC21	BPFS Lab	Kongming		BPFS	10	800	972.60	96.26	9,827.09
9	SC19	WekaIO	WekaIO on AWS	WekaIO	WekaIO Matrix	345	8,625	938.95	174.74	5,045.33
10	ISC20	TACC	Frontera	Intel	DAOS	60	1,440	763.80	78.31	7,449.56

IO500 SC21 10NODE LIST

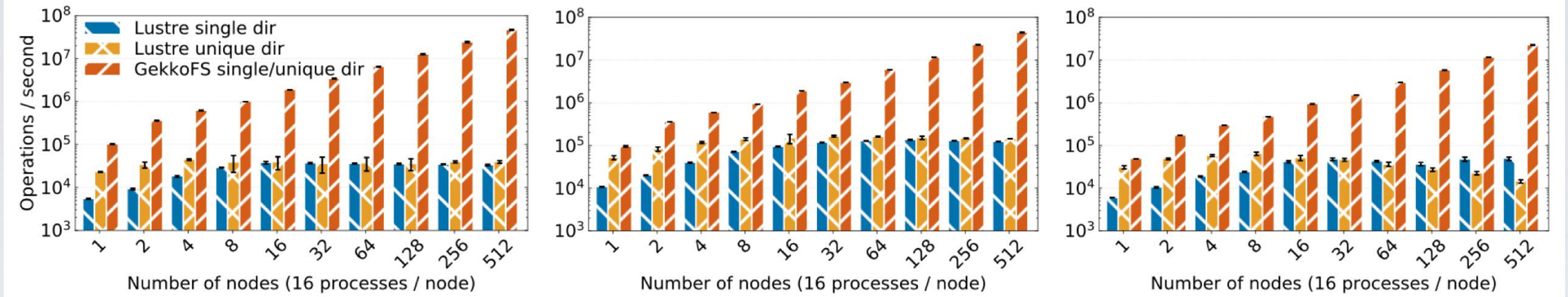
10 Node SC21 List

IO500 | 10 Node | Full | Historical | Customize | Download

This is the SC21 IO500 10-nodes list

# ↑	INFORMATION							IO500	
	BOF	INSTITUTION	SYSTEM	STORAGE VENDOR	FILE SYSTEM TYPE	CLIENT NODES	TOTAL CLIENT PROC.	SCORE ↑	BW (GIB/S)	MD (KIOP/S)
1	ISC21	Pengcheng Laboratory	Pengcheng Cloudbrain-II on Atlas 900	Pengcheng	MadFS	10	1,800	2,595.89	193.77	34,777.27
2	SC21	Huawei HPDA Lab	Athena	Huawei	OceanFS	10	1,720	2,395.03	314.56	18,235.71
3	SC21	Olympus Lab	OceanStor Pacific	Huawei	OceanFS	10	1,720	2,298.69	317.07	16,664.88
4	ISC21	Intel	Endeavour	Intel	DAOS	10	1,440	1,859.56	398.77	8,671.65
5	SC21	BPFS Lab	Kongming		BPFS	10	800	972.60	96.26	9,827.09
6	ISC20	Intel	Wolf	Intel	DAOS	10	420	758.71	164.77	3,493.56
7	ISC21	Lenovo	Lenovo-Lenox	Lenovo	DAOS	10	960	612.87	105.28	3,567.85
8	SC21	National Research Center of Tranlational Medicine at Shanghai Ruijin Hospital	ASTRA	NRCTM	DAOS	10	360	511.02	87.50	2,984.61
9	ISC20	TACC	Frontera	Intel	DAOS	10	420	508.88	79.16	3,271.49
10	ISC21	National Supercomputer Center in GuangZhou	Venus2	National Supercomputer Center in GuangZhou	kapok	10	480	474.10	91.64	2,452.87

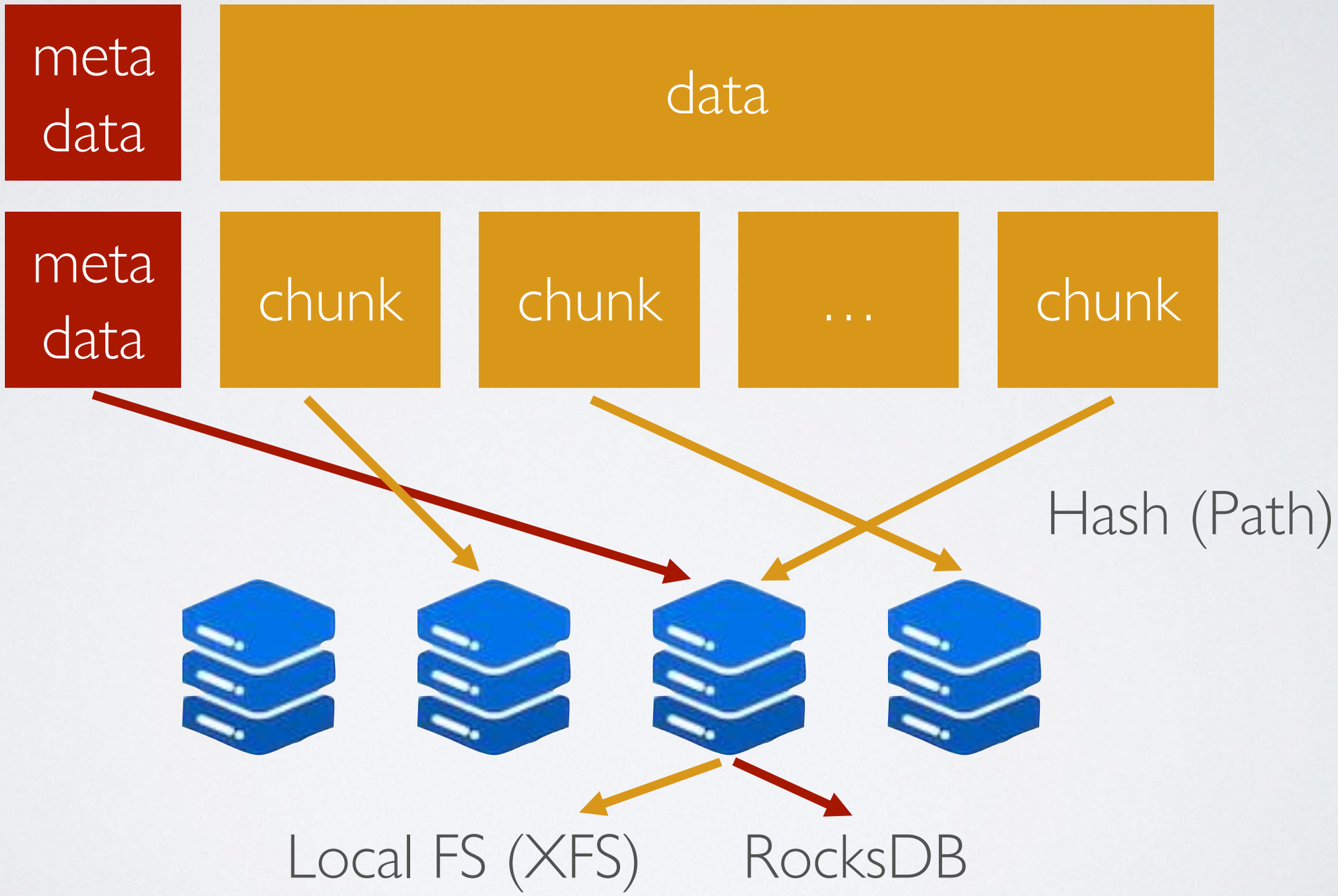
GekkoFS



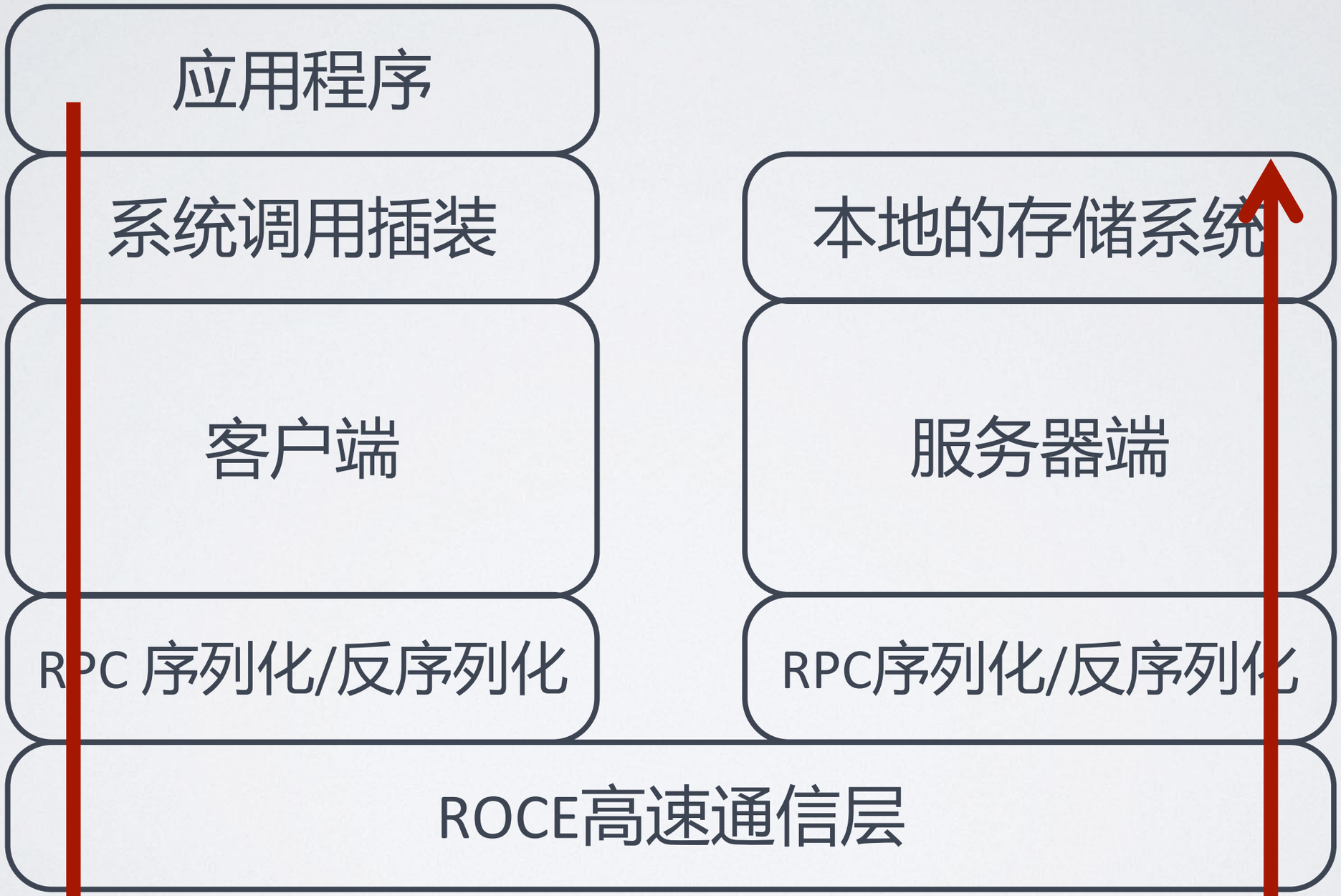
(a) Create throughput (b) Stat throughput (c) Remove throughput

- 大规模集群系统中具有良好的元数据性能
- 所依赖的库在ARM（鲲鹏）上非常的不稳定
- Libfabric 和华为网卡不兼容

MadFS元数据与数据分布策略



MadFS 体系结构


应用程序
系统调用插装
客户端
RPC 序列化/反序列化
本地的存储系统
服务器端
RPC序列化/反序列化
ROCE高速通信层

系统调用插装

- 用户层的系统调用插装库
  装载时重写所用进入系统调用的指令 *syscall* ➔ *jmp*
- DAOS开发为PMEM介质开发，GekkoFS中使用
- 仅仅支持 x86_64 体系结构，为ARM64重写

https://github.com/madsys-dev/syscall_intercept/tree/aarch64

```
65  intercept_asm_wrapper_tmpl:
66          # clone
67          cmp x8, #220
68          bne 0f
69          svc #0
70          b intercept_asm_wrapper_tmpl_end
71  0:
72          # save
73          stp     lr, x6, [sp, #-16]!
74  intercept_asm_wrapper_patch_desc_addr:
75          # load patch_desc address
76          movz x6, 0
77          movk x6, 0, lsl 16
78          movk x6, 0, lsl 32
79          movk x6, 0, lsl 48
80          # save patch_desc pointer on stack
81          str     x6, [sp, #-16]!
82  intercept_asm_wrapper_wrapper_level1_addr:
83          # load asm_wrapper address
84          movz x6, 0
85          movk x6, 0, lsl 16
86          movk x6, 0, lsl 32
87          movk x6, 0, lsl 48
88          # call asm_wrapper
89          blr x6
90          # restore
91          add sp, sp, #16
92          ldp lr, xzr, [sp], #16
93  intercept_asm_wrapper_tmpl_end:
94          /*
95           * This template must be appended here with a
96           * jump back to the intercepted code.
97           */
98          # b xxx
```

Template for each syscall

Leave 2 scratch regs

Fill in descriptor address

Fill in handler address

Fill in return address

```
mov x0, #1
svc #0 => b xxxxx
```

Rewrite syscall instructions

```
71  intercept_wrapper:
72      # Recover x6
73      ldr x6, [sp, #0x18]
74      # Down stack pointer
75      sub sp, sp, #0xe0
76      # Save caller-saved general registers
77      stp x0, x1, [sp, #0x00]
78      stp x2, x3, [sp, #0x10]
79      stp x4, x5, [sp, #0x20]
80      stp x6, x7, [sp, #0x30]
81      stp x8, x9, [sp, #0x40]
82      stp x10, x11, [sp, #0x50]
83      stp x12, x13, [sp, #0x60]
84      stp x14, x15, [sp, #0x70]
85      stp x16, x17, [sp, #0x80]
86      stp x18, x19, [sp, #0x90]
87      # Save caller-saved floating-point registers
88      stp d8, d9, [sp, #0xa0]
89      stp d10, d11, [sp, #0xb0]
90      stp d12, d13, [sp, #0xc0]
91      stp d14, d15, [sp, #0xd0]
92
93      mov x0, sp
94      mov x19, lr
95      bl intercept_routine
96      mov lr, x19
```

Save general regs

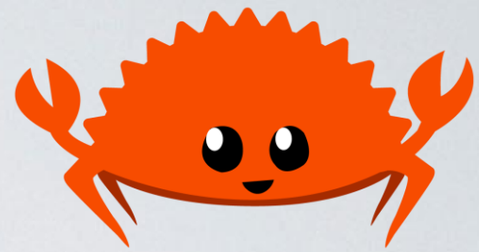
Save FP regs

Call C function

Restore regs…

MadFS 基础开发语言是Rust

为什么使用Rust



- 高性能：0开销的抽象
- 高可靠性：无段错误和数据竞争
- 开发高效：第三方库也非常稳定
- 对IO的支持：原生的异步IO，**async**hronous

协程支持

- 用户态的协程机制对于高速IO非常关键
- Rust支持了 **async-await** 语法
  - 使用类似同步的代码来编写异步的函数
  - 避免 C 语言中的各种回调函数
- Rust 社区有广泛使用的良好优化的异步运行时
  - Single-thread or multi-thread work-stealing scheduler

MadFS的通信层

- UCX

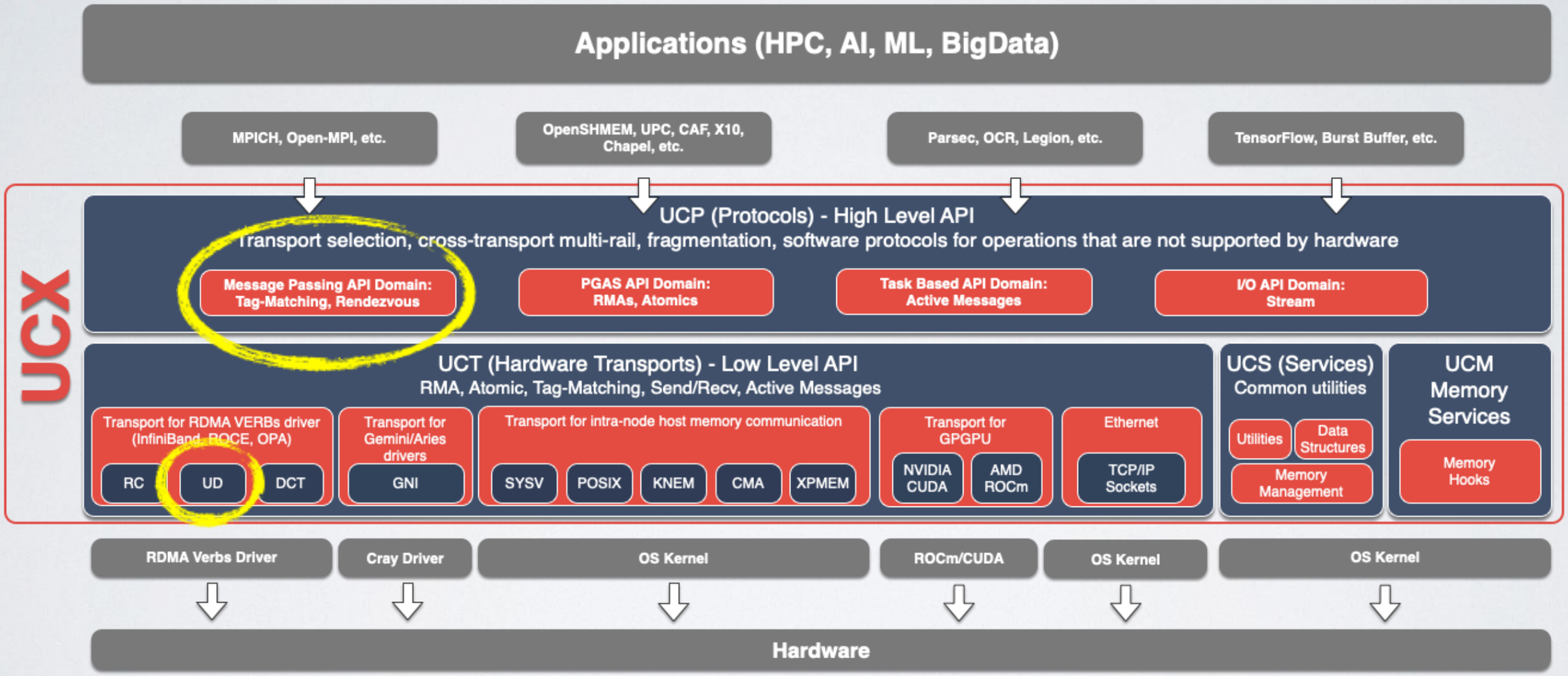
  华为网卡官方支持

- Written in C

  需要写一个Rust wrapper （使用异步模式）

https://github.com/madsys-dev/async-ucx

UCX


Applications (HPC, AI, ML, BigData)
MPICH, Open-MPI, etc.
OpenSHMEM, UPC, CAF, X10, Chapel, etc.
Parsec, OCR, Legion, etc.
TensorFlow, Burst Buffer, etc.
UCX
UCP (Protocols) - High Level API
Transport selection, cross-transport multi-rail, fragmentation, software protocols for operations that are not supported by hardware
Message Passing API Domain: Tag-Matching, Rendezvous
PGAS API Domain: RMAs, Atomics
Task Based API Domain: Active Messages
I/O API Domain: Stream
UCT (Hardware Transports) - Low Level API
RMA, Atomic, Tag-Matching, Send/Recv, Active Messages
Transport for RDMA VERBs driver (InfiniBand, ROCE, OPA)
RC
UD
DCT
Transport for Gemini/Aries drivers
GNI
Transport for intra-node host memory communication
SYSV
POSIX
KNEM
CMA
XPMEM
Transport for GPGPU
NVIDIA CUDA
AMD ROCm
Ethernet
TCP/IP Sockets
UCS (Services)
Common utilities
Utilities
Data Structures
Memory Management
UCM Memory Services
Memory Hooks
RDMA Verbs Driver
Cray Driver
OS Kernel
ROCm/CUDA
OS Kernel
OS Kernel
Hardware

RPC库

- **RPC库直接建立在 UCX之上**
- Use ***Tag Matching*** API for targeting (UCX)
- Use ***Vectored-IO*** for packing data and arguments (UCX)
- Use ***Serde*** framework for argument (de)serialization (Rust)
- Use ***FlexBuffers*** protocol for zero-copy deserialization (Rust)

Zero-copy 序列化与反序列化

```rust
#[derive(Debug, Serialize, Deserialize)]
pub enum RpcRequest<'a> {
    GetFsConfig,
    Statfs,
    Sync,
    Create {
        path: &'a str,
        mode: u32,
    },
    Remove {
        path: &'a str,
    },
    Stat {
        path: &'a str,
    },
    GetDirents {
        path: &'a str,
        seek: &'a str,
        len: usize,
    },
```

Reference

Avoid malloc, memcpy…

Meanwhile, keep it easy to use…

```rust
let mut s = flexbuffers::FlexbufferSerializer::new();
monster.serialize(&mut s).unwrap();
```

```rust
let reader = flexbuffers::Reader::get_root(header).unwrap();
let req = RpcRequest::deserialize(reader).unwrap();
```

Vectored IO

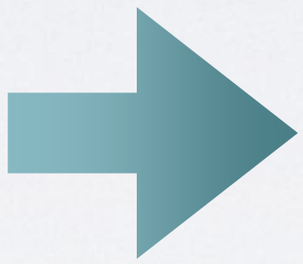
Client

Server

fn write(Data) {

Sz | Request

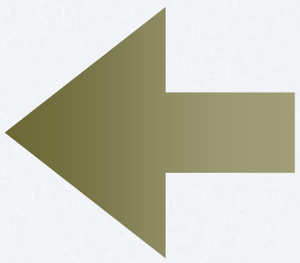
}

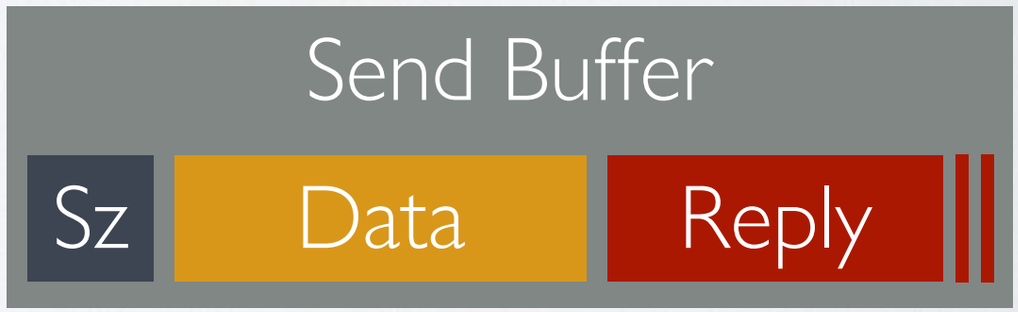


fn read(Data) {

Reply

}

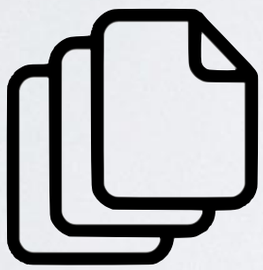




存储模型

元数据 ➔ RocksDB

{ full-path => metadata }

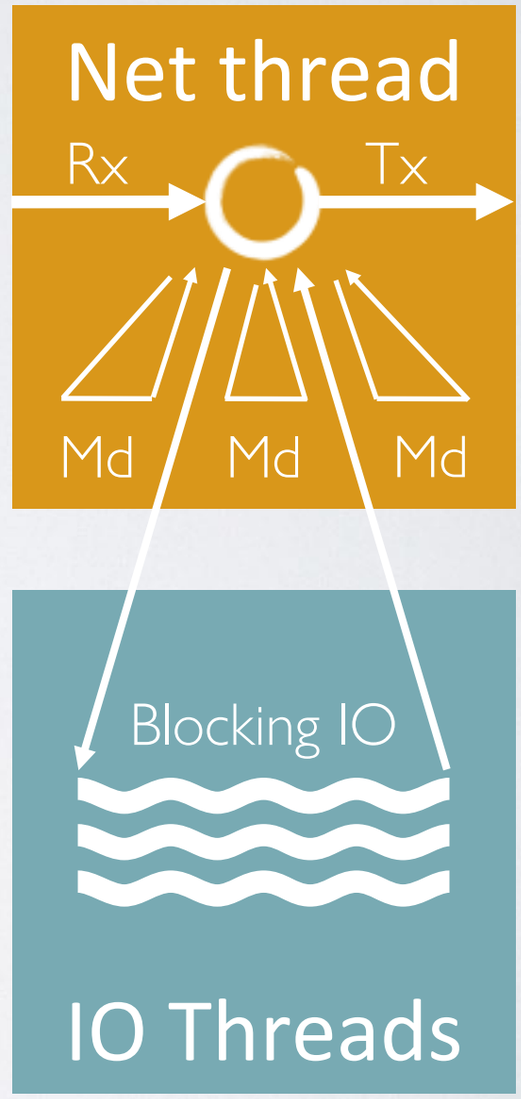


数据块 ➔ 本地文件系统

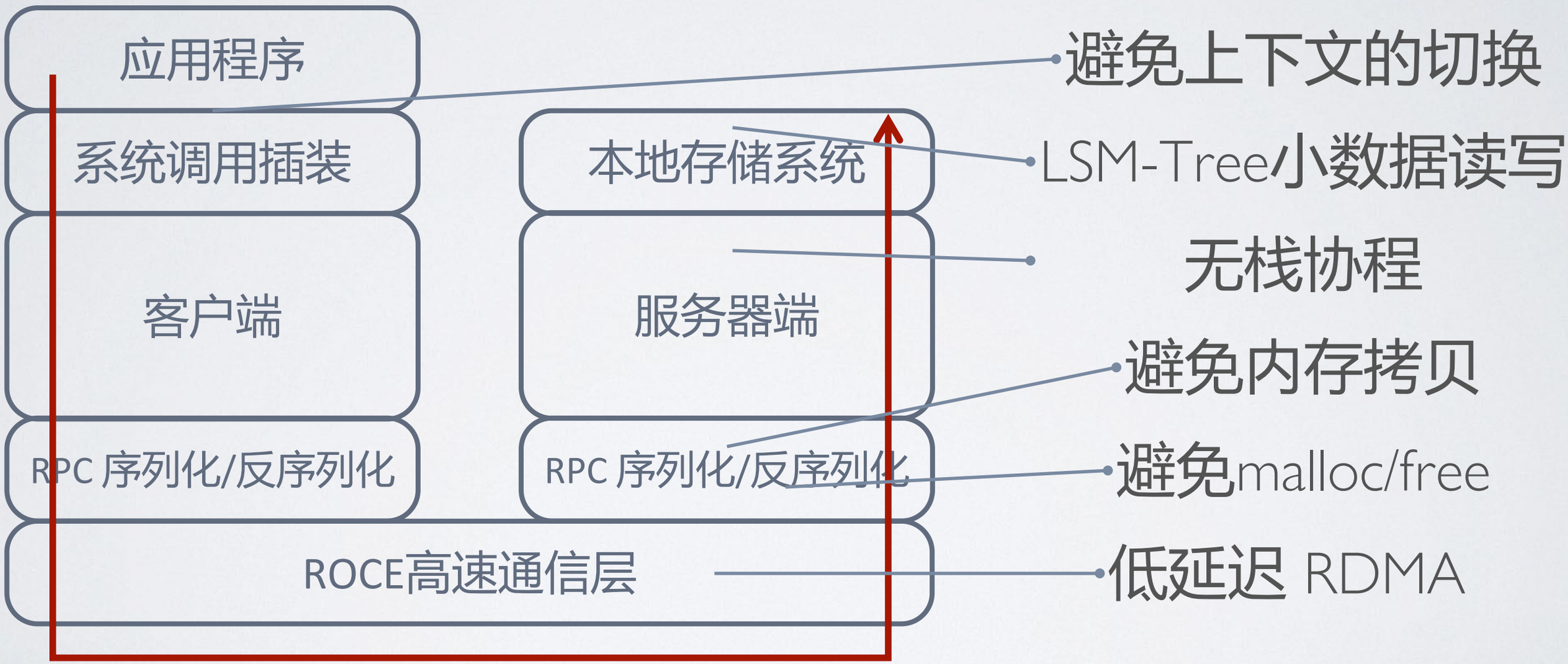
./<full-path>/<chunk-id> => data

线程模型

- 网络线程: 1 or 2
  - 使用Polling事件机制（ UCX worker ）
  - 16 RPC handler 协程
  - 元数据操作
  - Blocking ops给 IO 线程
- IO线程池的工作:
  - 文件读写；扫描RocksDB；处理Sync



性能优化小结


应用程序
系统调用插装
本地存储系统
客户端
服务器端
RPC 序列化/反序列化
RPC 序列化/反序列化
ROCE高速通信层
避免上下文的切换
LSM-Tree小数据读写
无栈协程
避免内存拷贝
避免malloc/free
低延迟 RDMA

SC20 Final Result

```
IO500 version io500-sc20
[RESULT]       ior-easy-write     1424.808689 GiB/s : time 564.166 seconds
[RESULT]    mdtest-easy-write    60169.632729 kIOPS : time 377.775 seconds
[RESULT]       ior-hard-write     1245.347896 GiB/s : time 467.116 seconds
[RESULT]    mdtest-hard-write     9453.618969 kIOPS : time 372.813 seconds
[RESULT]                 find    19359.924471 kIOPS : time 1350.331 seconds
[RESULT]        ior-easy-read     2157.710521 GiB/s : time 372.556 seconds
[RESULT]     mdtest-easy-stat    84376.480140 kIOPS : time 268.277 seconds
[RESULT]        ior-hard-read     1238.813319 GiB/s : time 466.851 seconds
[RESULT]     mdtest-hard-stat   106257.881498 kIOPS : time 33.077 seconds
[RESULT]   mdtest-easy-delete    61848.107522 kIOPS : time 365.974 seconds
[RESULT]     mdtest-hard-read    26439.064550 kIOPS : time 132.757 seconds
[RESULT]   mdtest-hard-delete    10115.188911 kIOPS : time 346.939 seconds
[SCORE] Bandwidth 1475.745858 GiB/s : IOPS 33622.191788 kiops : TOTAL 7043.991074
```

- Client: 255 nodes x 72 procs = 18,360 procs
- Server: 254 nodes x 12 procs = 3,048 procs

SC20 Final Result (10node)

```
IO500 version io500-sc20
[RESULT]       ior-easy-write      218.347788 GiB/s : time 309.612 seconds
[RESULT]    mdtest-easy-write    10096.610886 kIOPS : time 321.116 seconds
[RESULT]       ior-hard-write      164.761712 GiB/s : time 319.149 seconds
[RESULT]    mdtest-hard-write     1845.728978 kIOPS : time 319.937 seconds
[RESULT]                 find    13675.860737 kIOPS : time 278.587 seconds
[RESULT]        ior-easy-read      209.672740 GiB/s : time 322.418 seconds
[RESULT]     mdtest-easy-stat    24879.733021 kIOPS : time 129.545 seconds
[RESULT]        ior-hard-read      106.679098 GiB/s : time 492.905 seconds
[RESULT]     mdtest-hard-stat    18555.250866 kIOPS : time 31.643 seconds
[RESULT]   mdtest-easy-delete     8894.572633 kIOPS : time 362.347 seconds
[RESULT]     mdtest-hard-read     3594.165535 kIOPS : time 163.331 seconds
[RESULT]   mdtest-hard-delete     2891.539567 kIOPS : time 203.018 seconds
[SCORE] Bandwidth 168.425017 GiB/s : IOPS 7578.057274 kiops : TOTAL 1129.749719
```

- Client: 10 nodes x 144 procs = 1,440 procs
- Server: 50 nodes x 6 procs = 300 procs

SC20 => ISC21

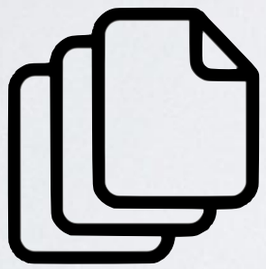
Optimize for:

• Memory Usage	Scale x2
• Local File IO and Fsync	BW +50%
• Small Files	IOPS +50%
• Find	Find x500, Score +50%

Small File Optimization

Metadata => RocksDB

{ full-path => metadata }



Chunks => Files on Local FS

./<full-path>/<chunk-id> => data

Chunk0 => RocksDB

Optimize for small files

Small File Optimization

	Base	OptSmall	
ior-easy-write	94.122	92.497	
mdtest-easy-write	5378.439	5422.553	
ior-hard-write	80.854	78.944	
mdtest-hard-write	463.743	1285.563	+177%
find	44604.009	36128.417	
ior-easy-read	91.108	90.295	
mdtest-easy-stat	10056.221	10951.796	
ior-hard-read	60.748	59.330	
mdtest-hard-stat	10638.537	9502.525	
mdtest-easy-delete	6216.329	6212.316	
mdtest-hard-read	2080.537	3446.449	+66%
mdtest-hard-delete	490.780	4127.369	+741%
Bandwidth	80.560	79.085	
IOPS	4071.717	6245.916	+53%
Score	572.729	702.823	+23%

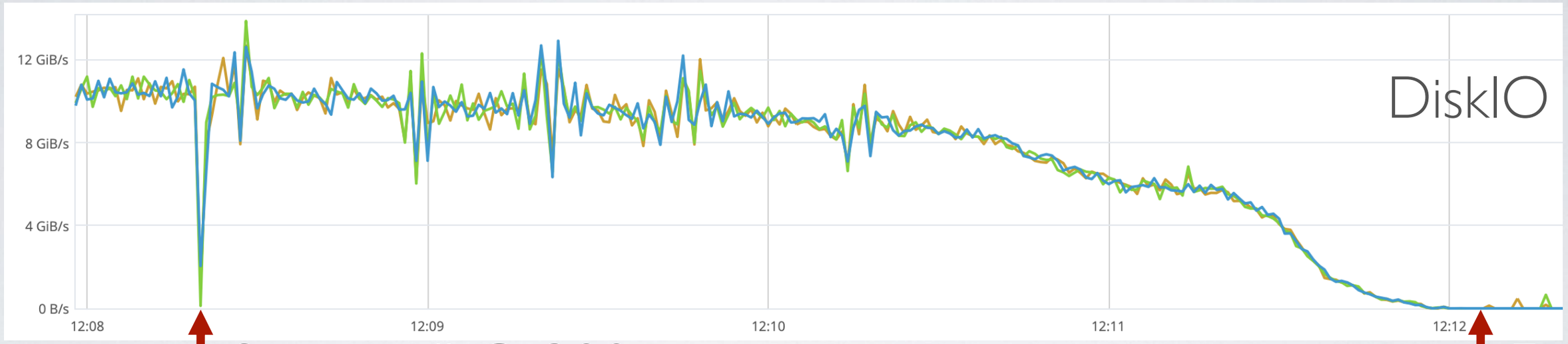
8 nodes x (18 server procs + 72 client procs), IO500 300s

The Long Tail Problem

Begin Stonewall @ 30s End

- Each client will issue an ***fsync*** after write complete
- The ***fsync*** request will be broadcast to each server
- On server, each ***fsync*** request triggers a global ***sync***
- Massive *sync* operations then block other writes

Handle Fsync



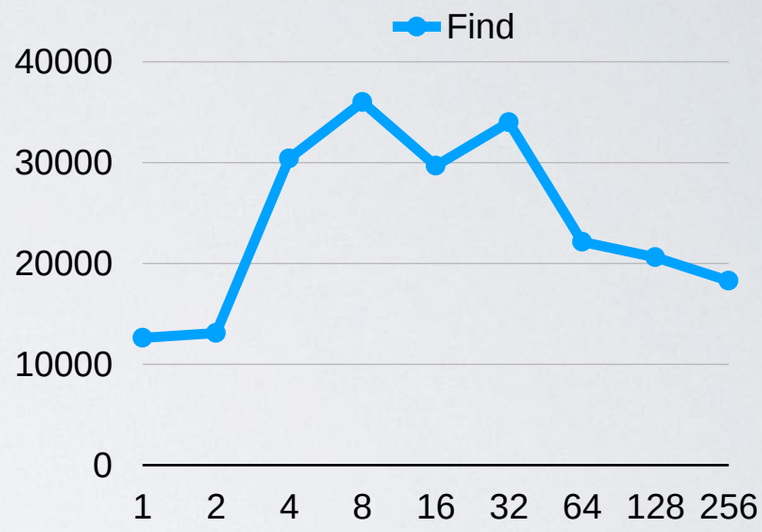
Stonewall @ 300s

End

- Spawn a background “sync” thread
- Each ***fsync*** request will notify it and wait for ***sync*** completion
- Separate ***fsync*** from other requests with different tags

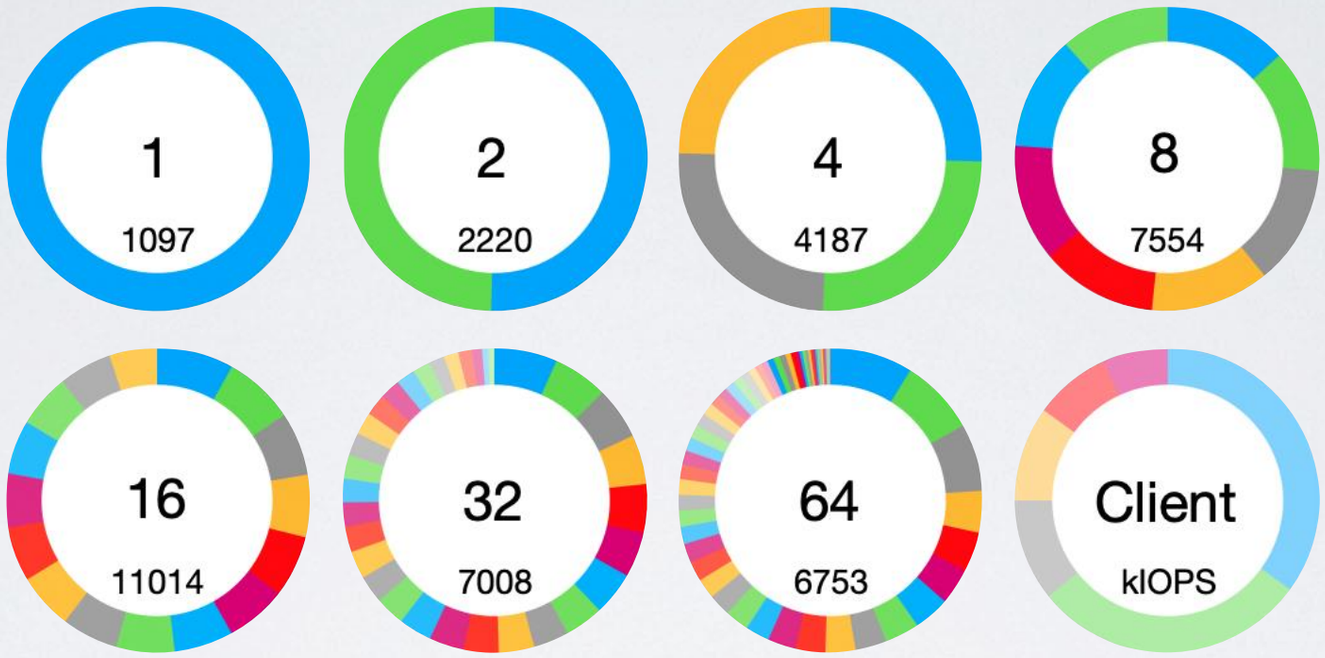
Find is NOT Scale

```
IO500 version io500-sc20
[RESULT]       ior-easy-write     1424.808689 GiB/s : time 564.166 seconds
[RESULT]    mdtest-easy-write    60169.632729 kIOPS : time 377.775 seconds
[RESULT]       ior-hard-write     1245.347896 GiB/s : time 467.116 seconds
[RESULT]    mdtest-hard-write     9453.618969 kIOPS : time 372.813 seconds
[RESULT]                 find    19359.924471 kIOPS : time 1350.331 seconds
[RESULT]          ior-easy-read     2157.710521 GiB/s : time 372.556 seconds
[RESULT]     mdtest-easy-stat    84376.480140 kIOPS : time 268.277 seconds
[RESULT]        ior-hard-read     1238.813319 GiB/s : time 466.851 seconds
[RESULT]     mdtest-hard-stat   106257.881498 kIOPS : time 33.077 seconds
[RESULT]   mdtest-easy-delete    61848.107522 kIOPS : time 365.974 seconds
[RESULT]     mdtest-hard-read    26439.064550 kIOPS : time 132.757 seconds
[RESULT]   mdtest-hard-delete    10115.188911 kIOPS : time 346.939 seconds
[SCORE] Bandwidth 1475.745858 GiB/s : IOPS 33622.191788 kiops : TOTAL 7043.991074
```

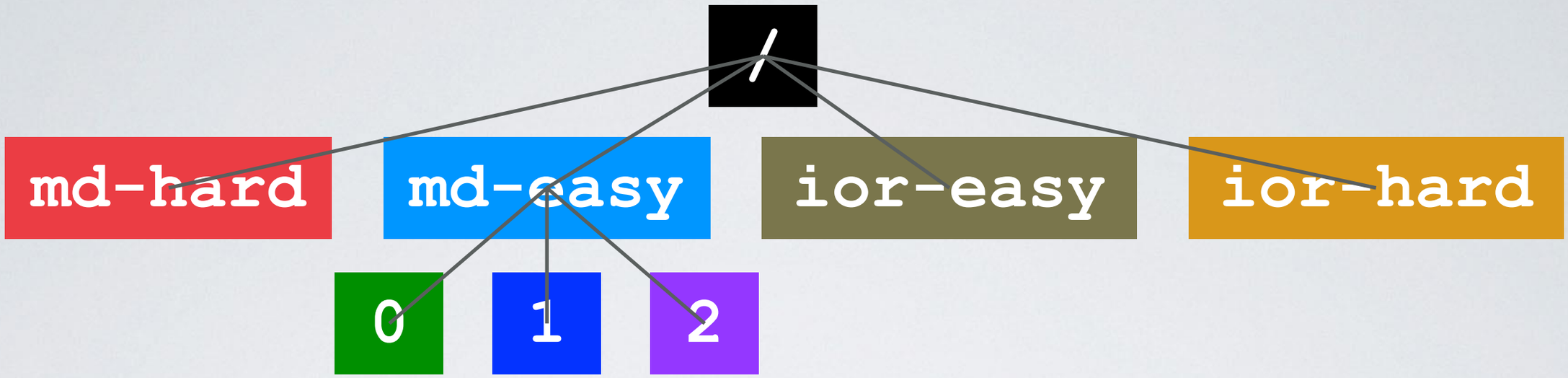


- mdtest-hard creates **~3,500,000,000** files in a **SINGLE** directory
- pfind can only scan them through **one** process

Load Imbalance



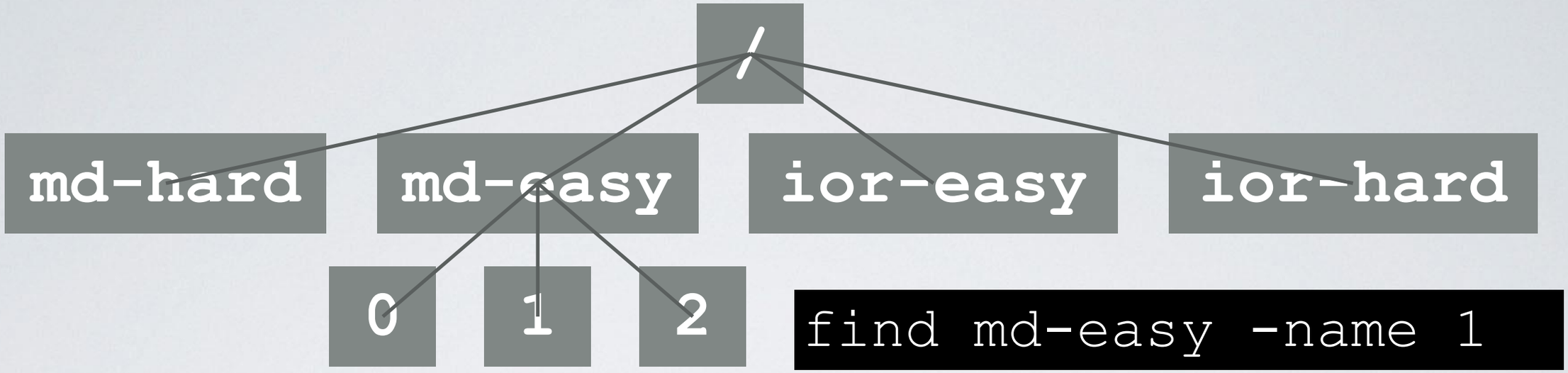
- pfind can do single-directory-parallel-access if the file system supports hash distribution of dentry cookie…
- We found that it still suffers from load imbalance problem

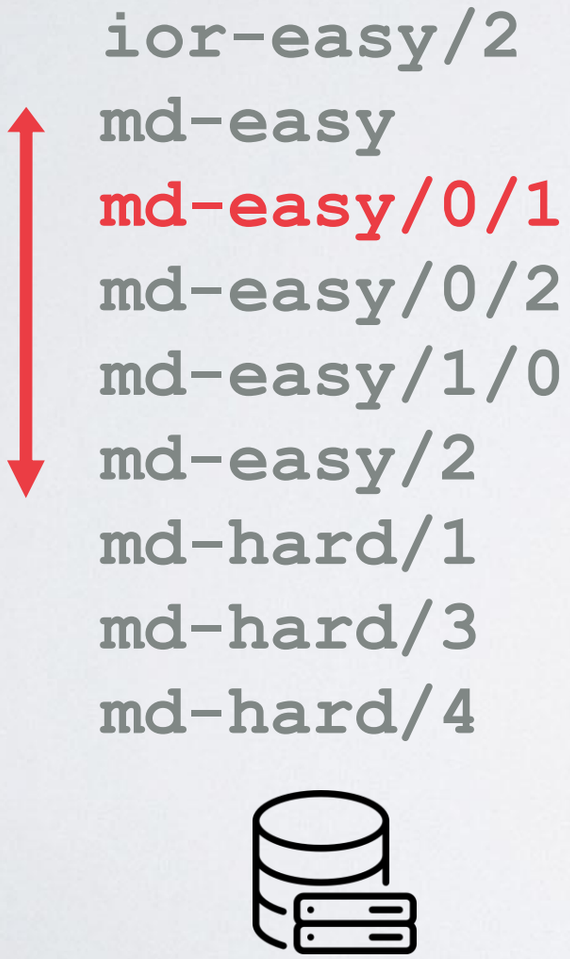


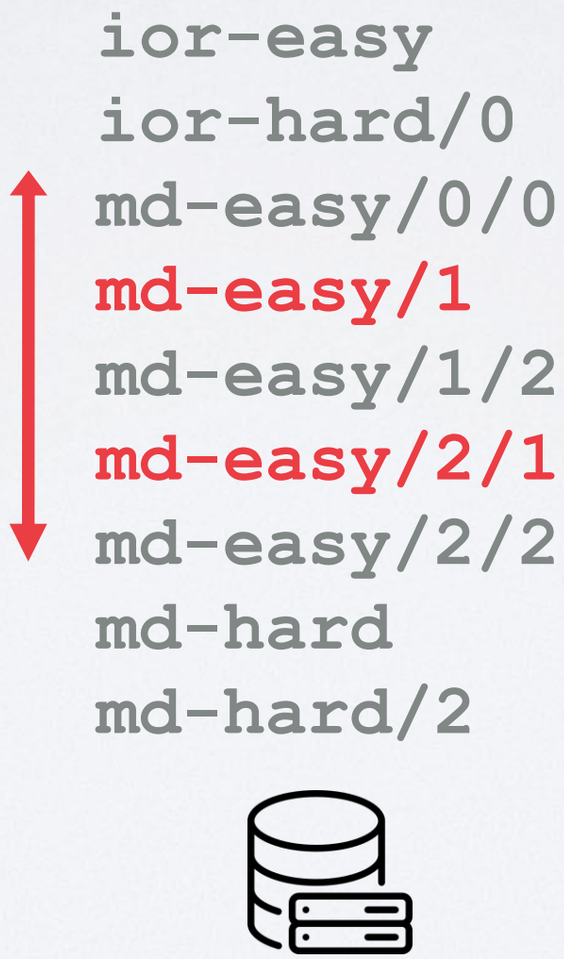
ior-easy/2
md-easy
md-easy/0/1
md-easy/0/2
md-easy/1/0
md-easy/2
md-hard/1
md-hard/3
md-hard/4

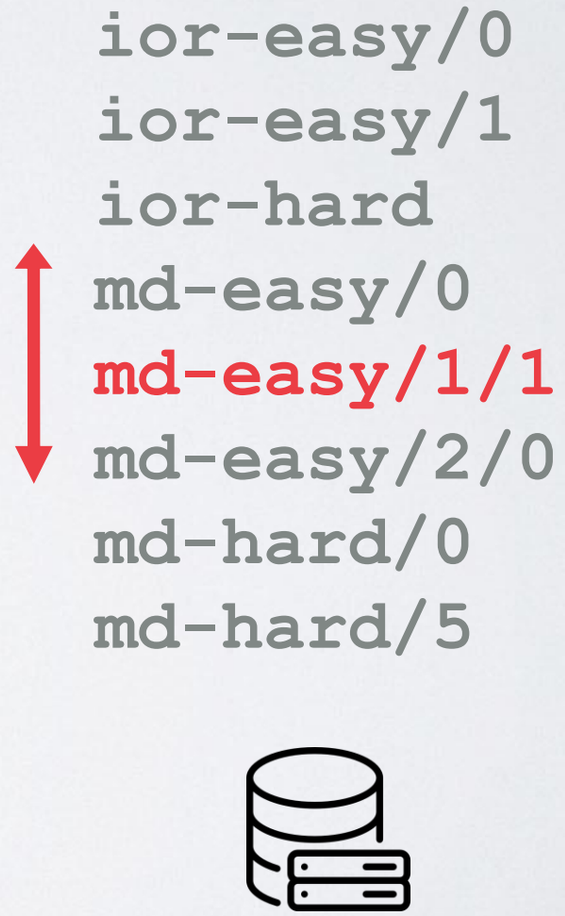
ior-easy
ior-hard/0
md-easy/0/0
md-easy/1
md-easy/1/2
md-easy/2/1
md-easy/2/2
md-hard
md-hard/2

ior-easy/0
ior-easy/1
ior-hard
md-easy/0
md-easy/1/1
md-easy/2/0
md-hard/0
md-hard/5


/
md-hard
md-easy
ior-easy
ior-hard
0
1
2
find md-easy -name 1


ior-easy/2
md-easy
md-easy/0/1
md-easy/0/2
md-easy/1/0
md-easy/2
md-hard/1
md-hard/3
md-hard/4


ior-easy
ior-hard/0
md-easy/0/0
md-easy/1
md-easy/1/2
md-easy/2/1
md-easy/2/2
md-hard
md-hard/2


ior-easy/0
ior-easy/1
ior-hard
md-easy/0
md-easy/1/1
md-easy/2/0
md-hard/0
md-hard/5

```
find md-easy -name 1
```

```
ior-easy/2
md-easy
md-easy/0/1
md-easy/0/2
md-easy/1/0
md-easy/2
md-hard/1
md-hard/3
md-hard/4
```

```
ior-easy
ior-hard/0
md-easy/0/0
md-easy/1
md-easy/1/2
md-easy/2/1
md-easy/2/2
md-hard
md-hard/2
```

```
ior-easy/0
ior-easy/1
ior-hard
md-easy/0
md-easy/1/1
md-easy/2/0
md-hard/0
md-hard/5
```

```
md-easy/0/1
md-easy/1
md-easy/2/1
md-easy/1/1
```

Madfs-Find


pfind
madfs-find
10000000
1000000
100000
10000
1
2
4
8
16
32
64
128
256
512
200x

ISC21 Final Result

```
IO500 version io500-isc21_v2 (standard)
[RESULT]       ior-easy-write     3360.298674 GiB/s : time 673.312 seconds
[RESULT]    mdtest-easy-write   278101.309437 kIOPS : time 318.025 seconds
[       ]            timestamp        0.000000 kIOPS : time 0.000 seconds
[RESULT]       ior-hard-write     3716.573255 GiB/s : time 448.352 seconds
[RESULT]    mdtest-hard-write   102419.383006 kIOPS : time 349.802 seconds
[RESULT]                 find  8991786.331160 kIOPS : time 13.770 seconds
[RESULT]        ior-easy-read     2803.716632 GiB/s : time 806.952 seconds
[RESULT]     mdtest-easy-stat   529803.484430 kIOPS : time 167.480 seconds
[RESULT]        ior-hard-read     3914.484508 GiB/s : time 415.836 seconds
[RESULT]     mdtest-hard-stat   410069.944303 kIOPS : time 88.234 seconds
[RESULT]   mdtest-easy-delete   287724.351211 kIOPS : time 307.527 seconds
[RESULT]     mdtest-hard-read   209906.918749 kIOPS : time 171.290 seconds
[RESULT]   mdtest-hard-delete   183148.570395 kIOPS : time 196.161 seconds
[SCORE ] Bandwidth 3421.624277 GiB/s : IOPS 396872.816141 kiops : TOTAL 36850.368552
```

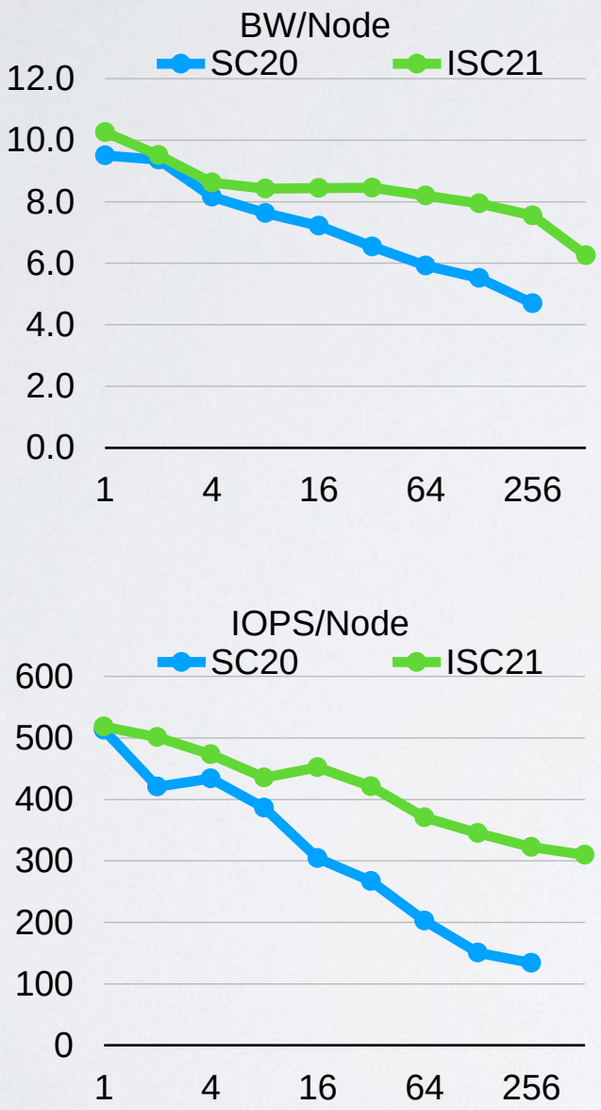
- Client: 512 nodes x 72 procs = 36,864 procs
- Server: 512 nodes x 24 procs = 12,288 procs

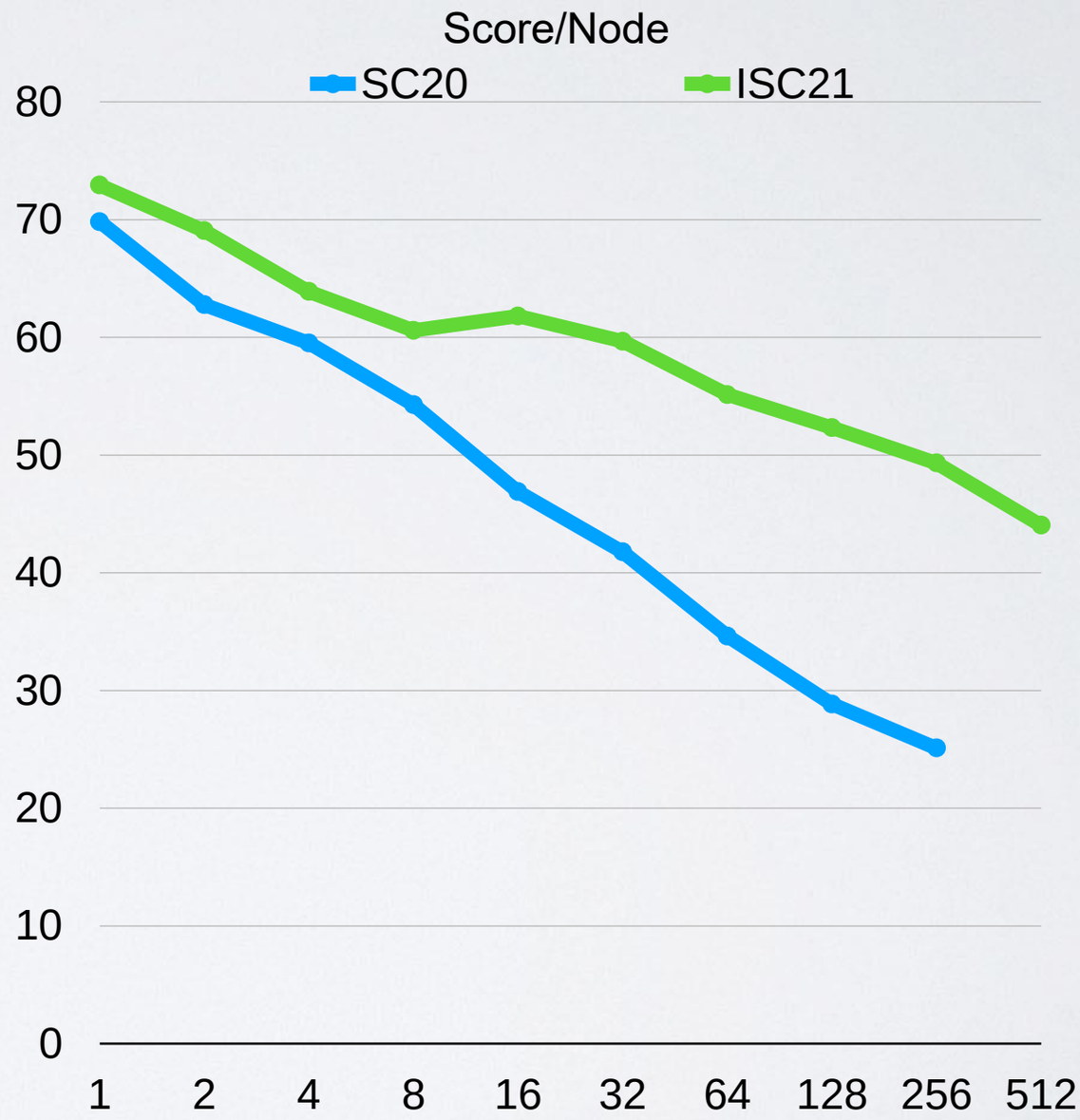
ISC21 Final Result (10node)

```
IO500 version io500-isc21_v2 (standard)
[RESULT]       ior-easy-write      222.987510 GiB/s : time 305.831 seconds
[RESULT]    mdtest-easy-write    19176.710145 kIOPS : time 333.678 seconds
[      ]            timestamp        0.000000 kIOPS : time 0.000 seconds
[RESULT]       ior-hard-write      197.272000 GiB/s : time 318.301 seconds
[RESULT]    mdtest-hard-write     8494.136829 kIOPS : time 327.979 seconds
[RESULT]                 find  1584644.889026 kIOPS : time 5.778 seconds
[RESULT]        ior-easy-read      149.939929 GiB/s : time 455.657 seconds
[RESULT]     mdtest-easy-stat    36333.268045 kIOPS : time 176.582 seconds
[RESULT]        ior-hard-read      213.722239 GiB/s : time 293.621 seconds
[RESULT]     mdtest-hard-stat    35438.079693 kIOPS : time 79.381 seconds
[RESULT]   mdtest-easy-delete    26920.996932 kIOPS : time 237.966 seconds
[RESULT]     mdtest-hard-read    15692.363037 kIOPS : time 177.994 seconds
[RESULT]   mdtest-hard-delete    15239.971832 kIOPS : time 183.249 seconds
[SCORE ] Bandwidth 193.766302 GiB/s : IOPS 34777.267068 kiops : TOTAL 2595.893380
```

- Client: 10 nodes x 180 procs = 1,800 procs
- Server: 50 nodes x 24 procs = 1,200 procs

Scalability Improvement





THANKS

MadFS is built on top of:

- Rust
- UCX
- Syscall-intercept
- RocksDB & rust-rocksdb
- Tokio
- Serde & FlexBuffers
- ……

The IO500 test was made possible by the cooperation of:

- Pengcheng Laboratory
- Huawei Technologies
- Tsinghua University