CERN

Evaluating object drives and non-volatile memory

EOS

Disk Storage @ CERN

Andreas-Joachim Peters

for the EOS project and IT-DSS

IT Data & Storage Service Group

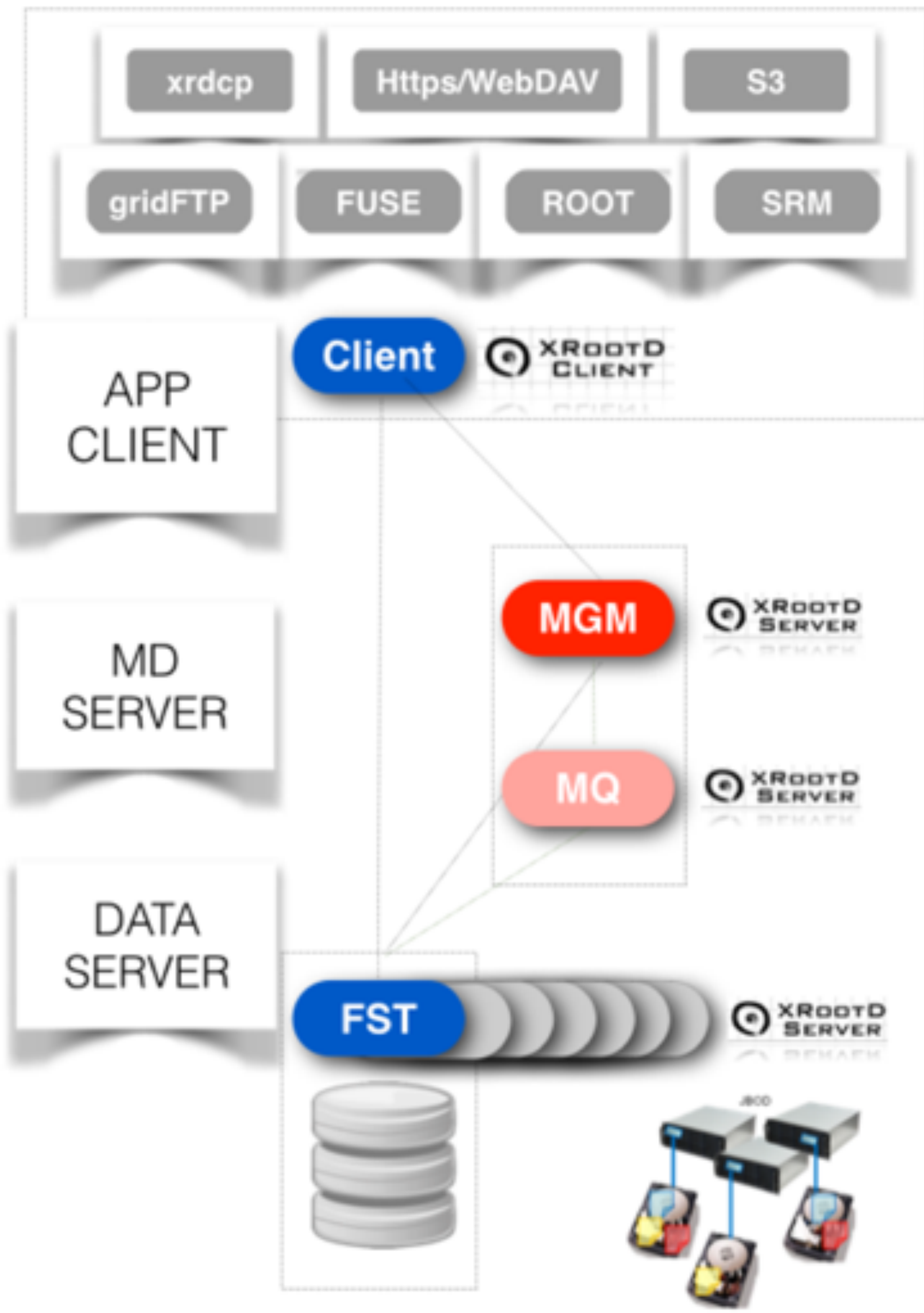
NEC’2015 1.10.2015

Andreas.Joachim.Peters@cern.ch

Overview

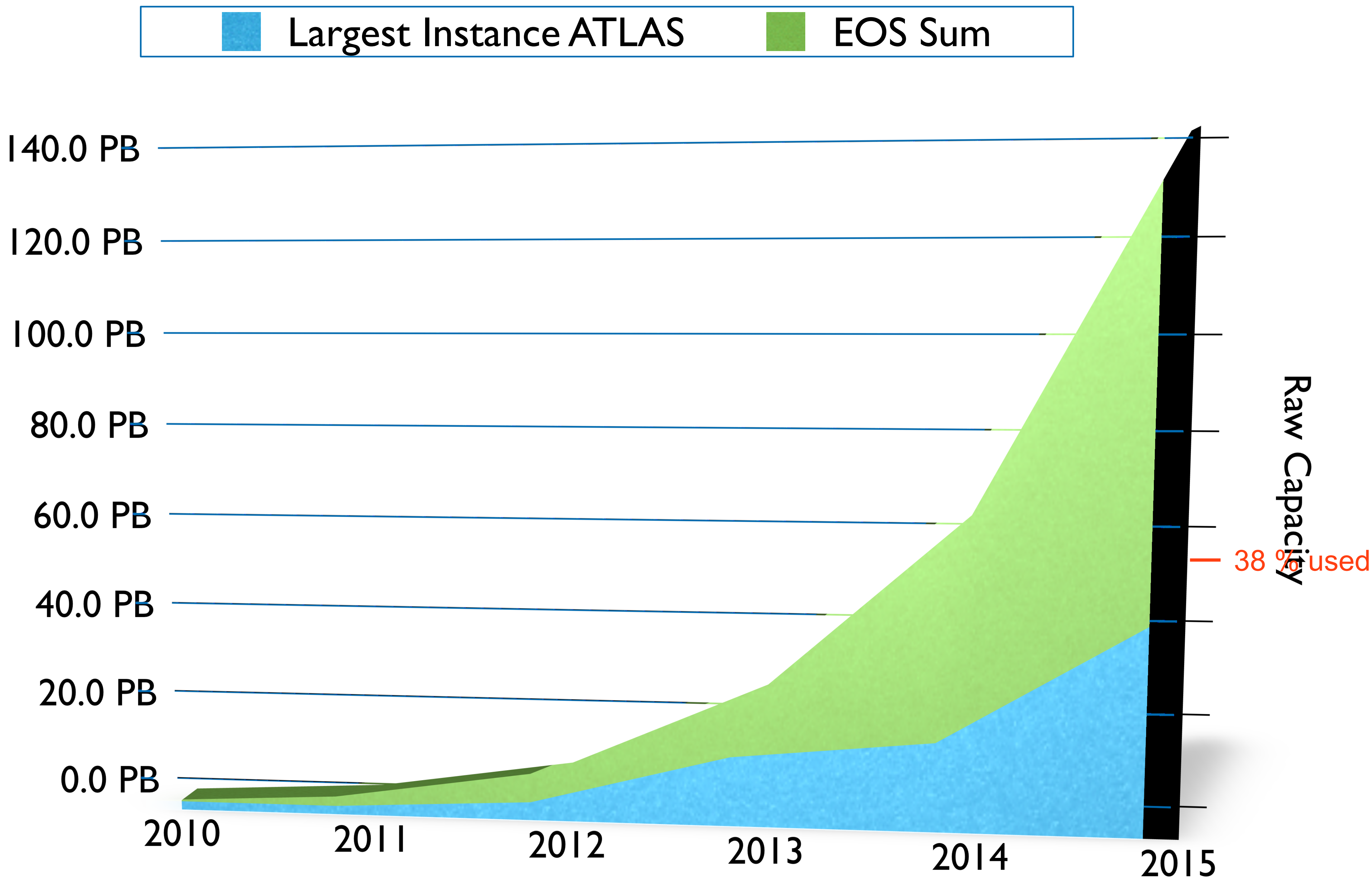
- Introduction to EOS
- Open Kinetic
- NVRAM R&D

Architecture



- project since 2010
- production since 2011
- simple - GPL - JBOD hardware
- in-memory namespace
- strong security
  [server side security]
- many protocols
- quota, tunable QoS
- Dev&Ops @ CERN/IT

EOS DEPLOYMENT



EOS STORAGE IN NUMBERS

April 2015	
Capacity	**140 PB**
Server	**1.400**
Hard Disks	**44k**
Files	**271 M**
Directories	**26 M**
Replicas	**0.5 B**
Connectivity [theor.]	**13 Tbit**
random IOPS	**2.2 M**
Disk BW [theor.]	**3.3 TB/s**
Internal Messaging	**150 kHz**
State Machine	**3M kv pairs**
Users storing data	**~3k**
Quota rules	**9.600**

single thread namespace stat rate
160 kHz

multi threaded namespace stat rate
1 MHz

memory footprint **0.5-1 kb/file**

Flat View is a scalability limitation!

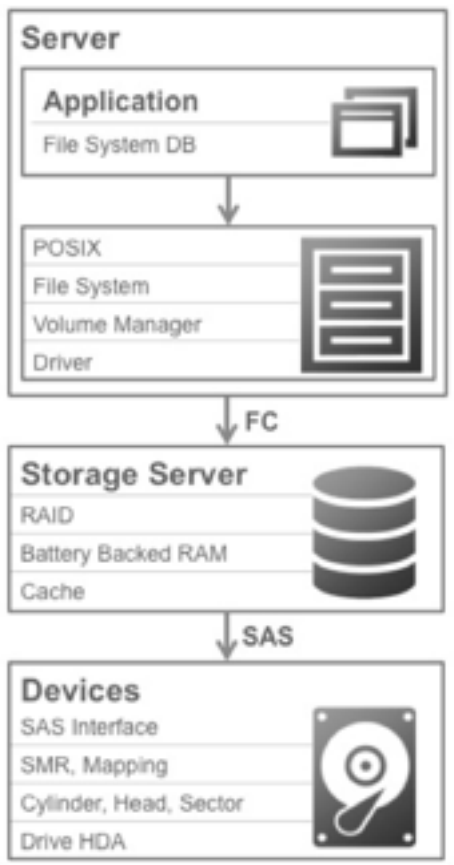
Seagate
Open Kinetic

ethernet drives

Open Kinetic

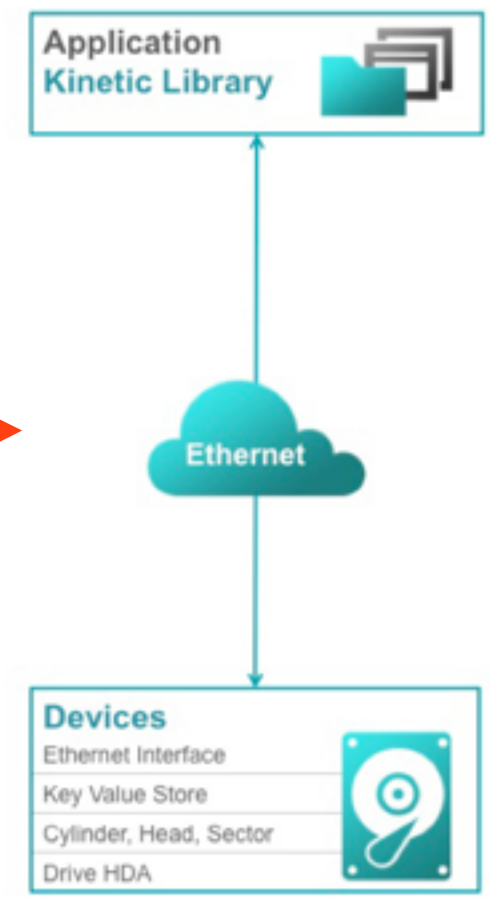
for software defined storage

conventional storage system



- POSIX

kinetic open storage platform



- Kinetic API

Kinetic more interesting than ever …

> Storage controllers, SANs and even file systems are all under threat from Kinetic, an extraordinarily disruptive storage architecture.

SEATTLE, LinuxCon/CloudOpen/ContainerCon— August 17, 2015 — The Linux Foundation, the nonprofit organization dedicated to accelerating the growth of Linux and collaborative development, today announced a new effort to define and promote open source software and standards for cloud object storage technologies.

The new Collaborative Project is the Kinetic Open Storage Project and includes founding members Cisco, Cleversafe, Dell, Digital Sense, Huawei, NetApp, Open vStorage, Red Hat, Scality, Seagate, SwiftStack, Toshiba and Western Digital.

Open Kinetic API

- Kinetic API
  - Access Control
    - READ - can read
    - WRITE - can write
    - DELETE - can delete
    - RANGE - can do range
    - SETUP - can setup device
    - P2POP - can do p2p copy
    - GETLOG - can get log
    - SECURITY - can set security
  - NOOP - like ping
  - PUT - store object max. value size 1 MB
  - DELETE - delete object
  - FLUSH - flush outstanding PUT/DELETE to device (=sync)
  - GET - retrieve value + meta data
  - GETVERSION - retrieve version tag for object
  - GETNEXT - return next sorted key
  - GETPREVIOUS - return previous sorted key
  - GETKEYRANGE - return keys in range
  - SETCLUSTERVERSION - set cluetser version
  - SETPIN - instant secure erase
  - SECURITY - set ACL
  - GETLOG - retrieve log
  - PEERTOPEERPUSH - copy KV between drives

- API less feature rich than rados API - low-level
  - no partial value get/updates/append - only full object GET/PUT
  - no arbitrary map per object, but vector clock/version
  - no clustering support between devices, but P2P push
- protocol implemented with google protocol buffers
- disk uses sorted string tables and log structured merge tree technology

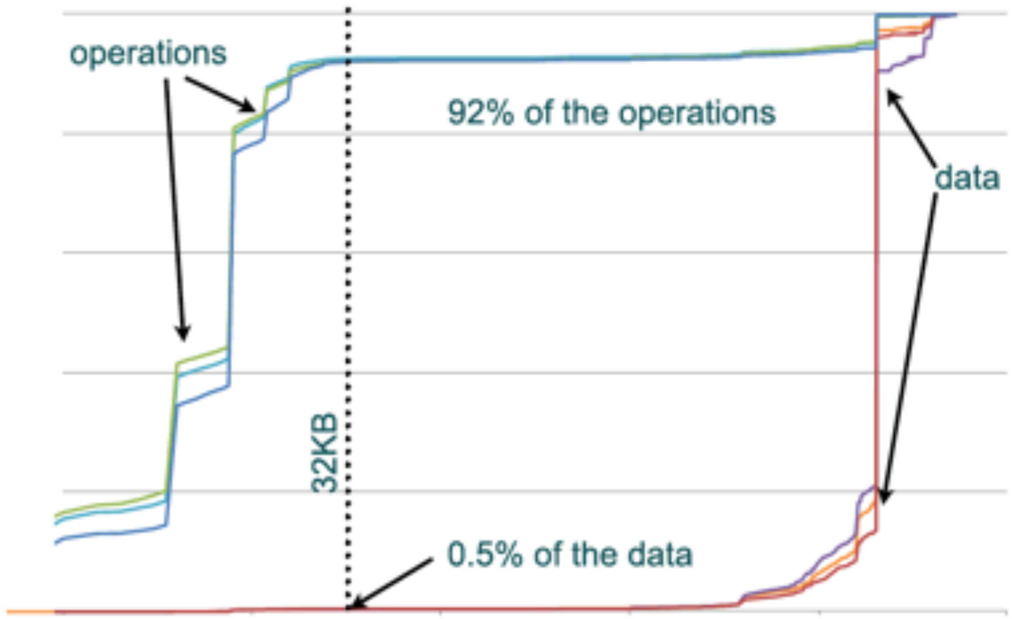
- need to implement high-level API & clustering software : **libkineticio**

Seagate Kinetic

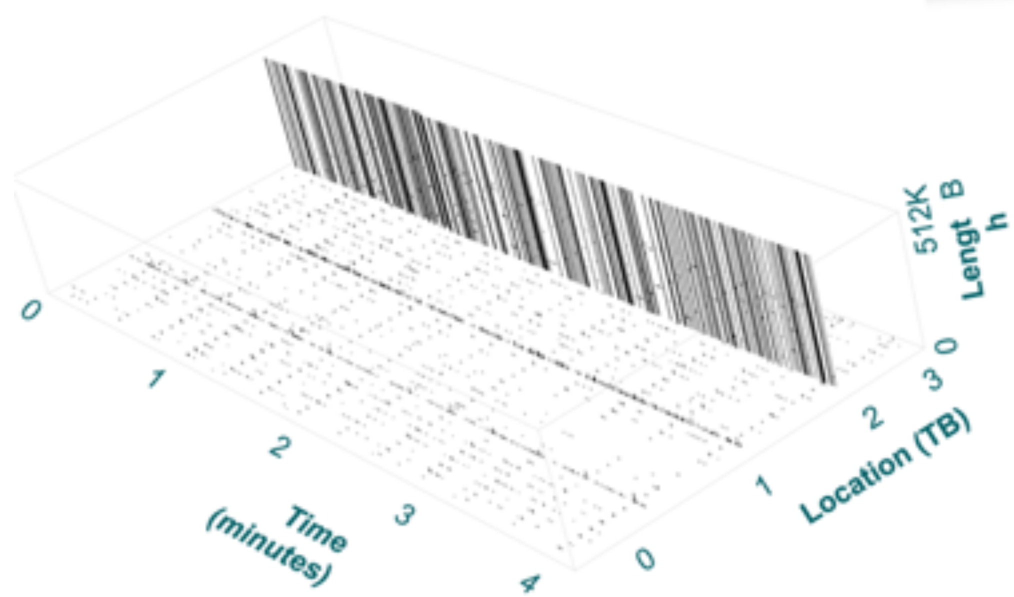
- why kinetic technology?
  - **fits technology of shingled disks**
  - **better random write**
  - **less meta-data overhead**
  - **lower TCO**

- performance expectation
  - **random/sequential write, sequential read: 50 MB/s for 1M objects**
  - **random read -15% to traditional drives**
  - **~ 1000 random write OOPS**

- integrated by
  - **swift**
    - access via gateways
  - **ceph** not provided



example of traditional IO inefficiency



Open Kinetic

Why and how to integrate them in EOS?

- Kinetic concept has potentially simple(r) deployment concept
  - install - register MAC - remote config - operation
  - required top of the rack switches and ports identical to conventional disk server
  - no disk-server association anymore
  - no visible Linux OS

- exploit Kinetic technology in a way that EOS does not need to manage individual drives anymore
  - HA clusters of Kinetic drives
  - downscale storage leaf nodes by e.g. 256
    - today we manage 14k in largest instance
    - tomorrow we would manage only 55 Kinetic Cluster

Seagate Kinetic

Installed Kinetic Cluster in CERN CC

21x12-Disk (4TB) SuperMicro Kinetic Server

1 PB usable capacity

- each server provides internal switch with 2x 1GBit Unit Uplink
- 40 GBit Rack Uplink

Open Kinetic

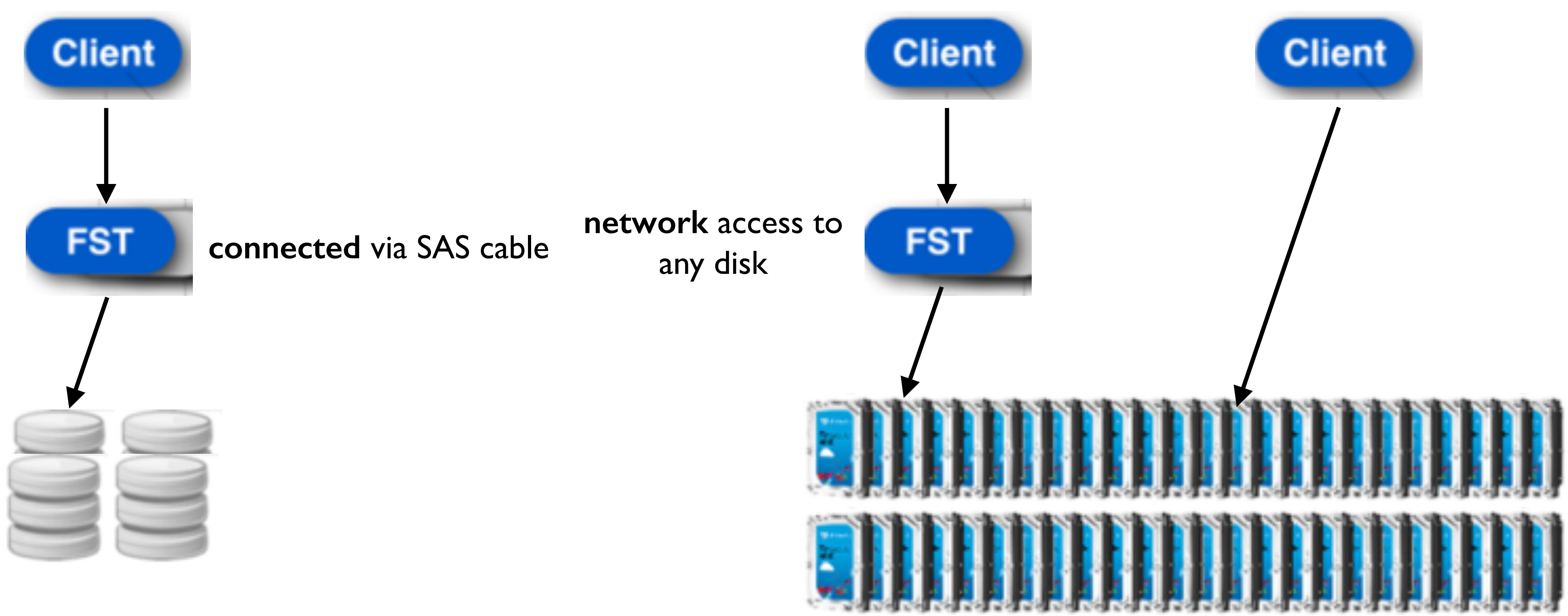
EOS Software Architecture

Old Style

New style

SAS Model
Single Server provides access to attached disks

Kinetic Model
Proxy Server(s) provide access to disks
Client may have direct access to Kinetic disks



Open Kinetic

libkineticio - IO for clustered kinetic drives

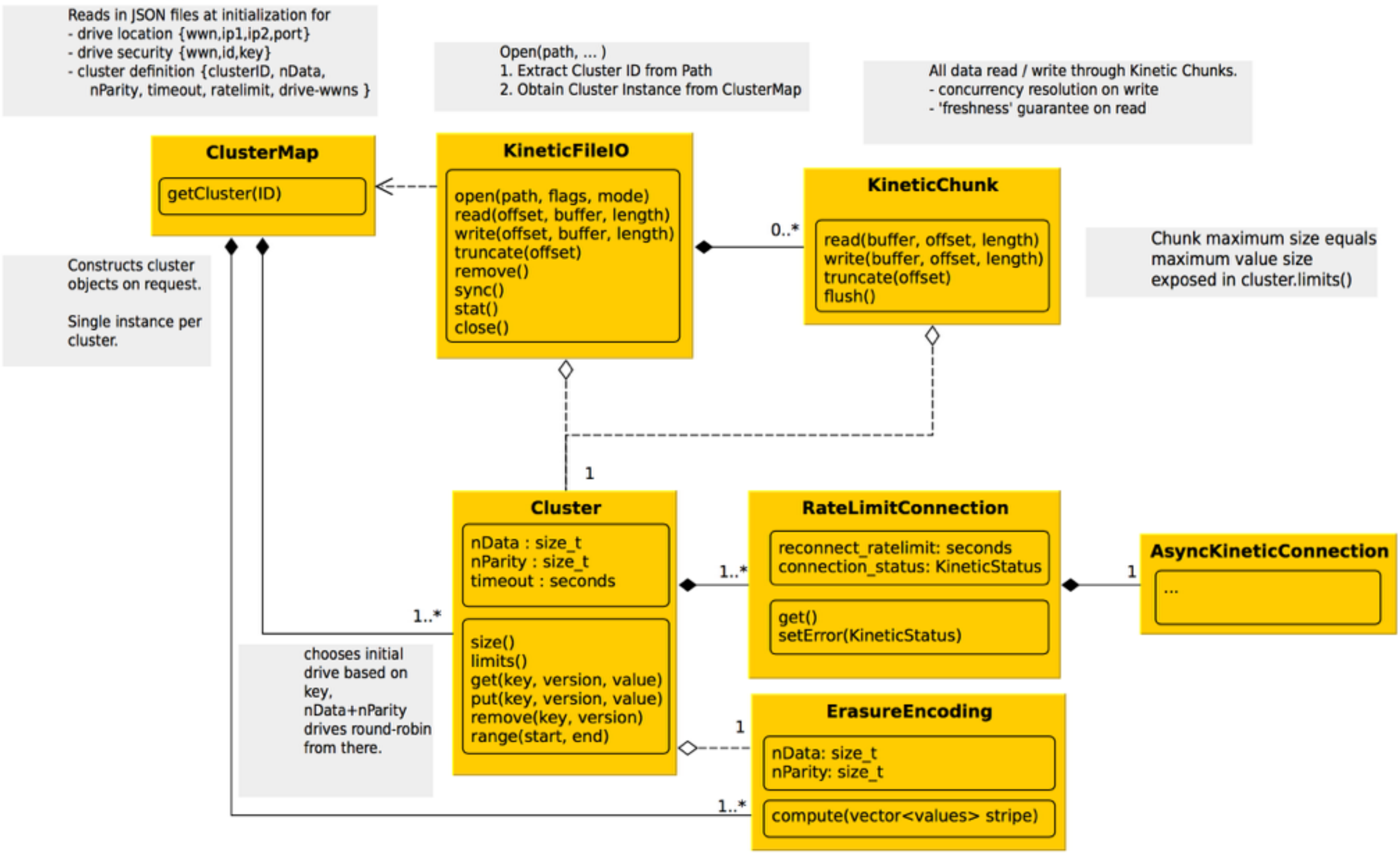
- development of **libkineticio** in C++11 Developer: Paul Lensing
  - provides parallel IO for chunking files over a drive cluster
  - provides file meta data KV interface
  - provides HA via Intel's ISA Erasure Encoding libarry
    - reconstrucion, hinted handoff …
- development of EOS **console tools** for kinetic administration
  - cluster configuration (k,m)
  - cluster consistency scrubbing
  - cluster repair

DRAFT

eos fs kinetic-status <id>
eos fs kinetic-repair <id>
eos fs kinetic-list
eos fs kinetic-config <id> timeout / reconnect
eos fs kinetic-setup <id> nData / nParity / subchunk-size

Open Kinetic

libkineticio - IO for clustered kinetic drives

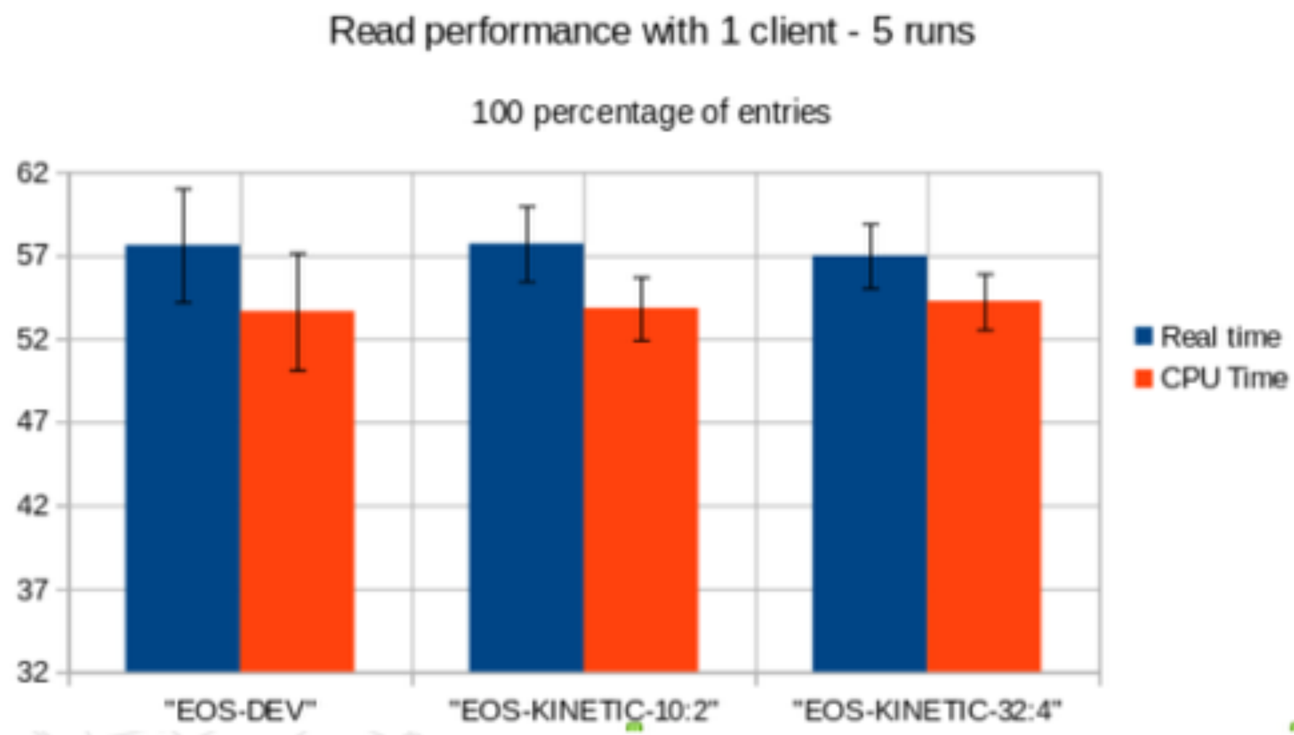


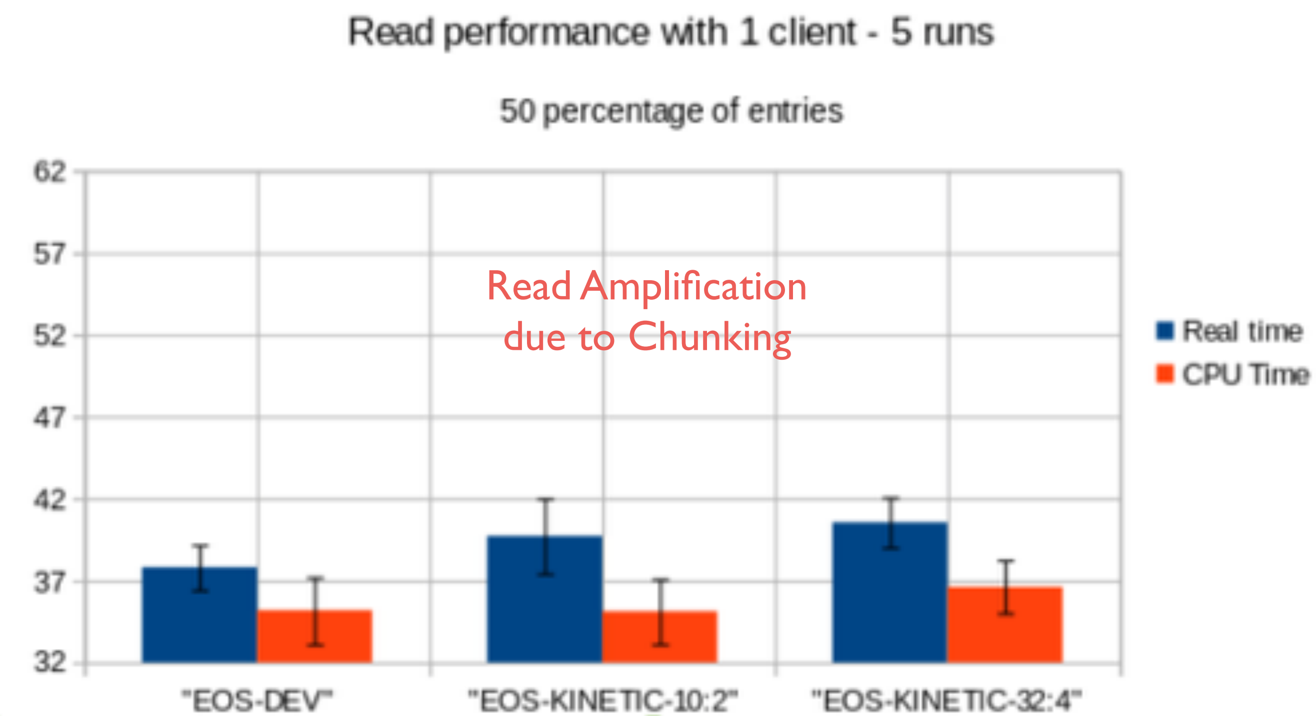
- Test with 10GE FST gateway/server comparing
  - conventional disk server 35 disks
  - Kinetic Cluster 42 disks & (10,2) EC configuration
  - Kinetic Cluster 42 disks & (32,4) EC configuration

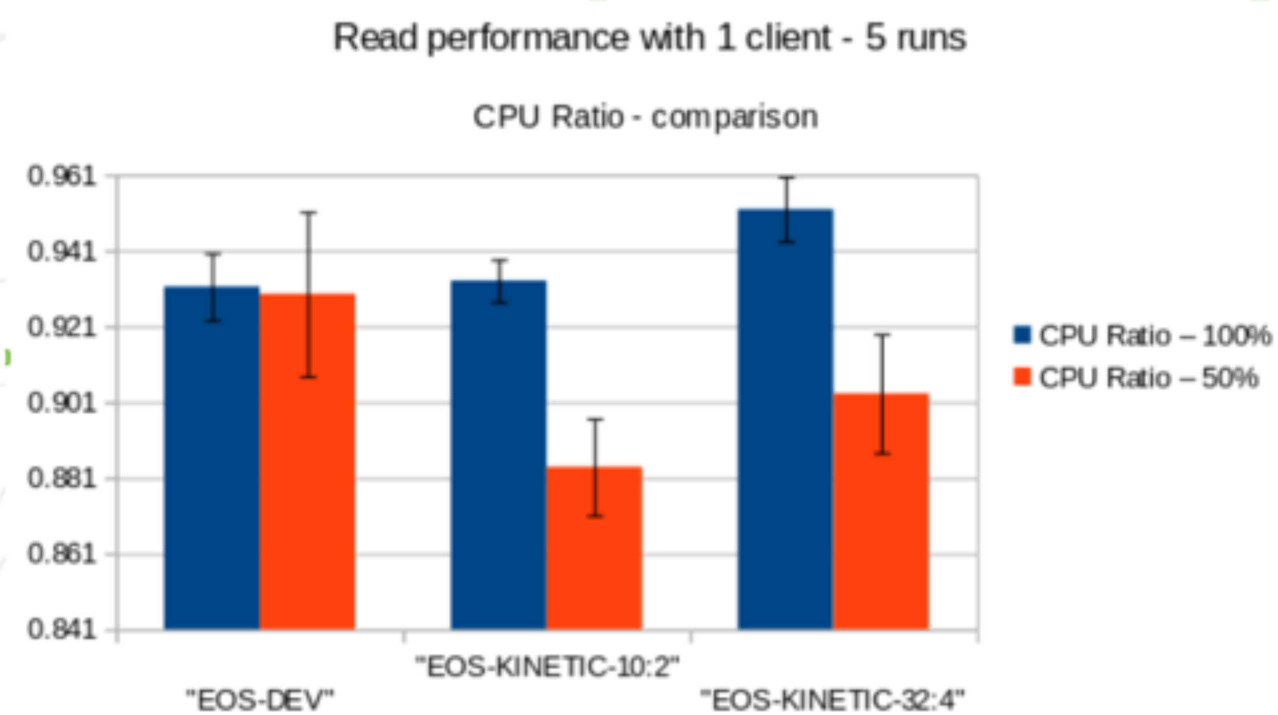
Seagate Kinetic

ROOT TTree Analaysis

Tests performed by: Ivana Petya







Seagate Kinetic

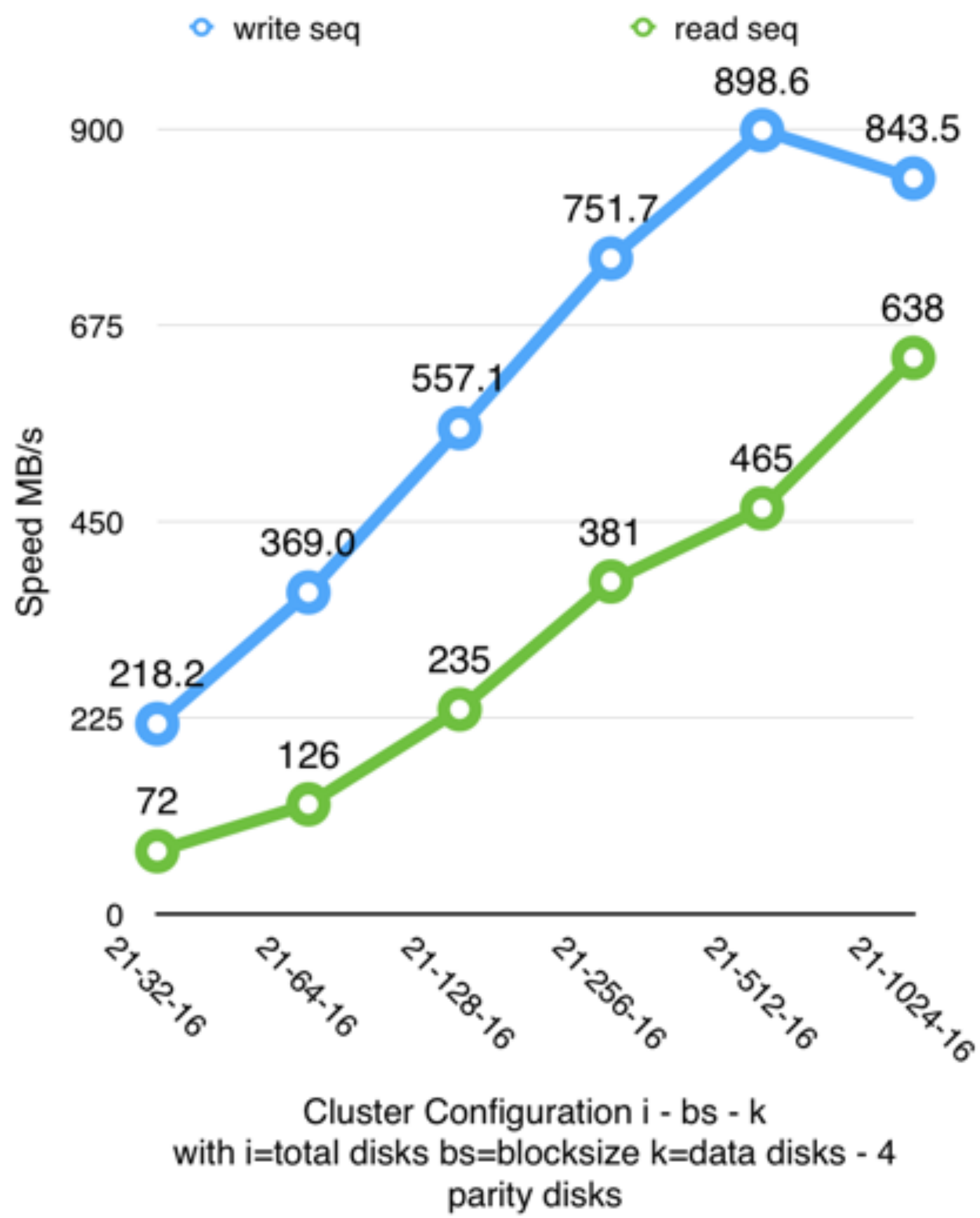
Blocksize Impact

each Chunk is split over m disks `blocksize=chunksize/m`

Kinetic Cluster Benchmark varying the block size

	write seq	read seq
21-32-16	218.24	72.32
21-64-16	368.96	126.08
21-128-16	557.12	234.88
21-256-16	751.68	381.44
21-512-16	898.56	465.28
21-1024-16	843.52	637.76

RS: m=16 k=4



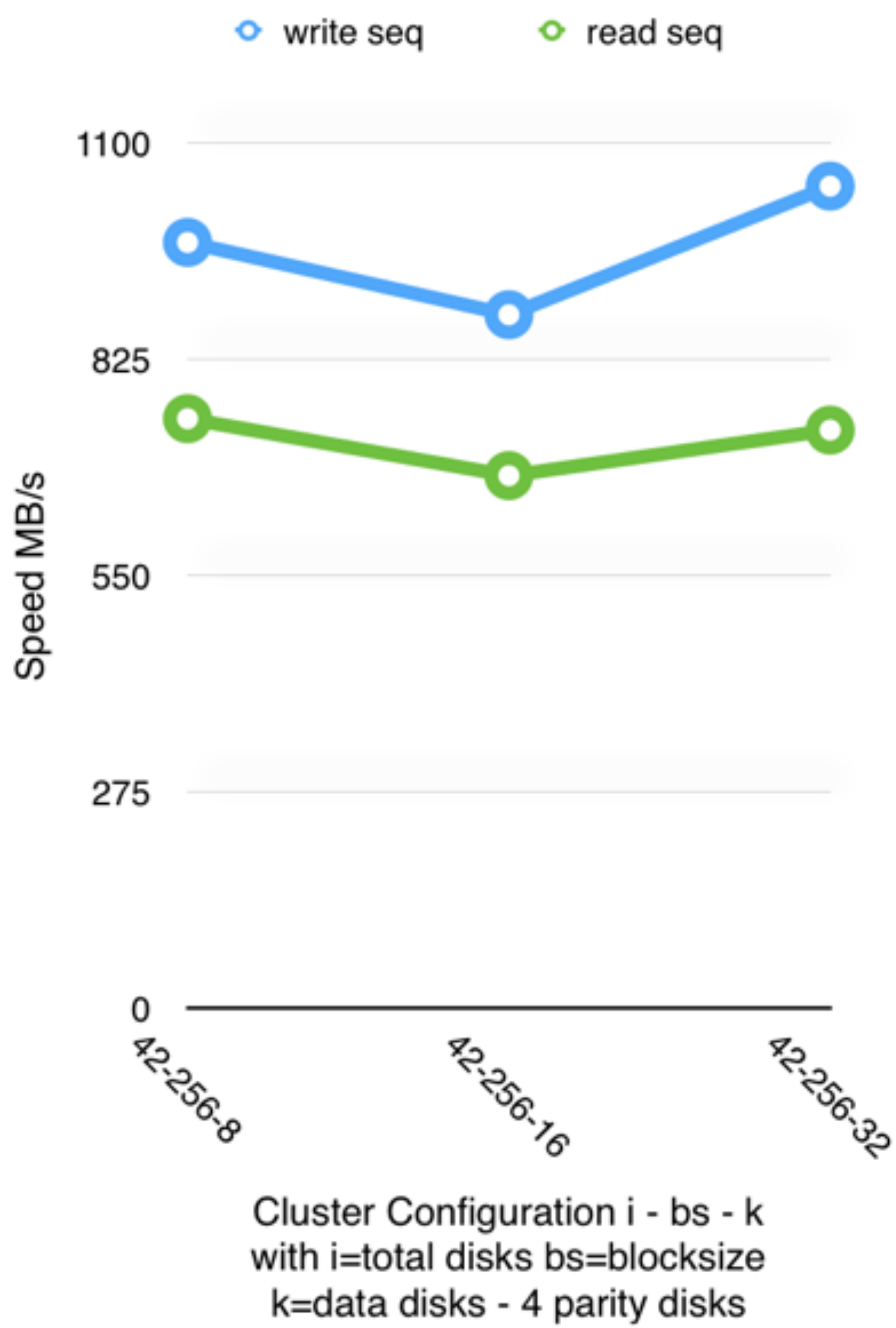
Seagate Kinetic

Data Drive Number Impact

Kinetic Cluster Benchmark varying the number of data disks

	write seq	read seq
42-256-8	973.248	749.216
42-256-16	881.28	676.8
42-256-32	1044.8	734.72

RS: m=<var> k=4



Seagate Kinetic

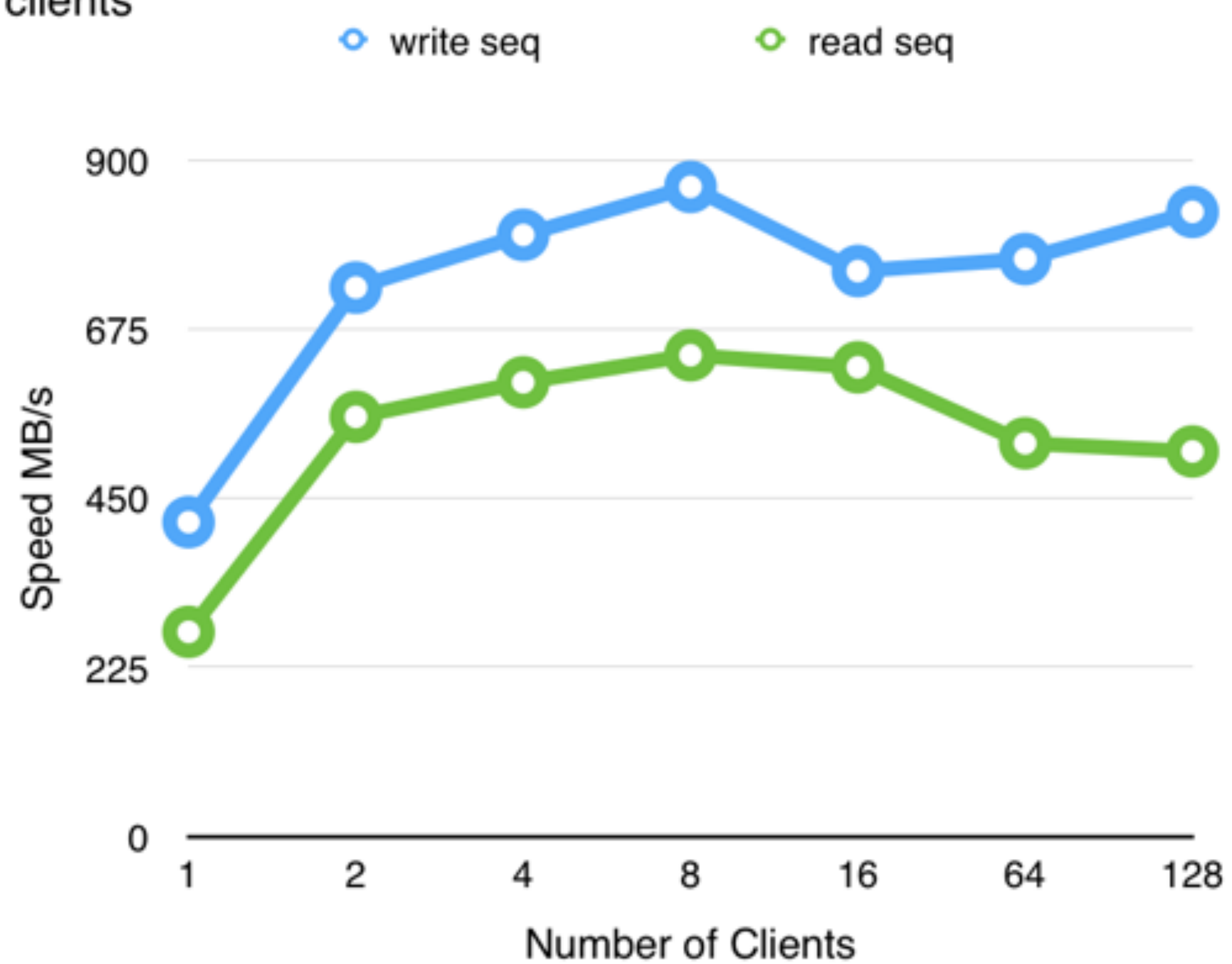
Client Scaling

- Kinetic Cluster 45 disks & (16,2) EC configuration

Kinetic Cluster Benchmark varying the number of clients

21-1024-16	write seq	read seq
1	418	272
2	730	558
4	800	604
8	864	640
16	752	624
64	768	522.88
128	830.72	512

RS: m=16 k=4
bs=1MB #disks=21



NVRAM for EOS

Client Scaling

CERN targets openlab partnership with **Data Storage Institute** in Singapore to evaluate NVRAM technology as persistency model for an in-memory namespace used in EOS

Idea: non-volative memory avoids boot time of the EOS namespace because meta-data structures used by the MGM application can be kept persistency in memory

Summer Student Project of Marti Bosch

- Simulated by DIMM RAM with a battery
- more sophisticated technologies incoming
- EOS used as a ‘testbed’ for further use
- Persistency is a ‘vertical’ property
    - transactional updates for consistency
    - persistent memory should not point to non-persistent memory

Mnemosyne[1] exposes persistency to C/C++:

- **pstatic** variables are stored persistently
- **pmalloc/pfree** allocate persistent memory
- **persistent** annotations ensure correctness
- **atomic blocks** allow transaction control

[1] Volos, Tack and Swift (2011)

Mnemosyne - Programming Example

Simplified example:

(courtesy of Sergio Ruocco and Le Duy Khan, DSI)

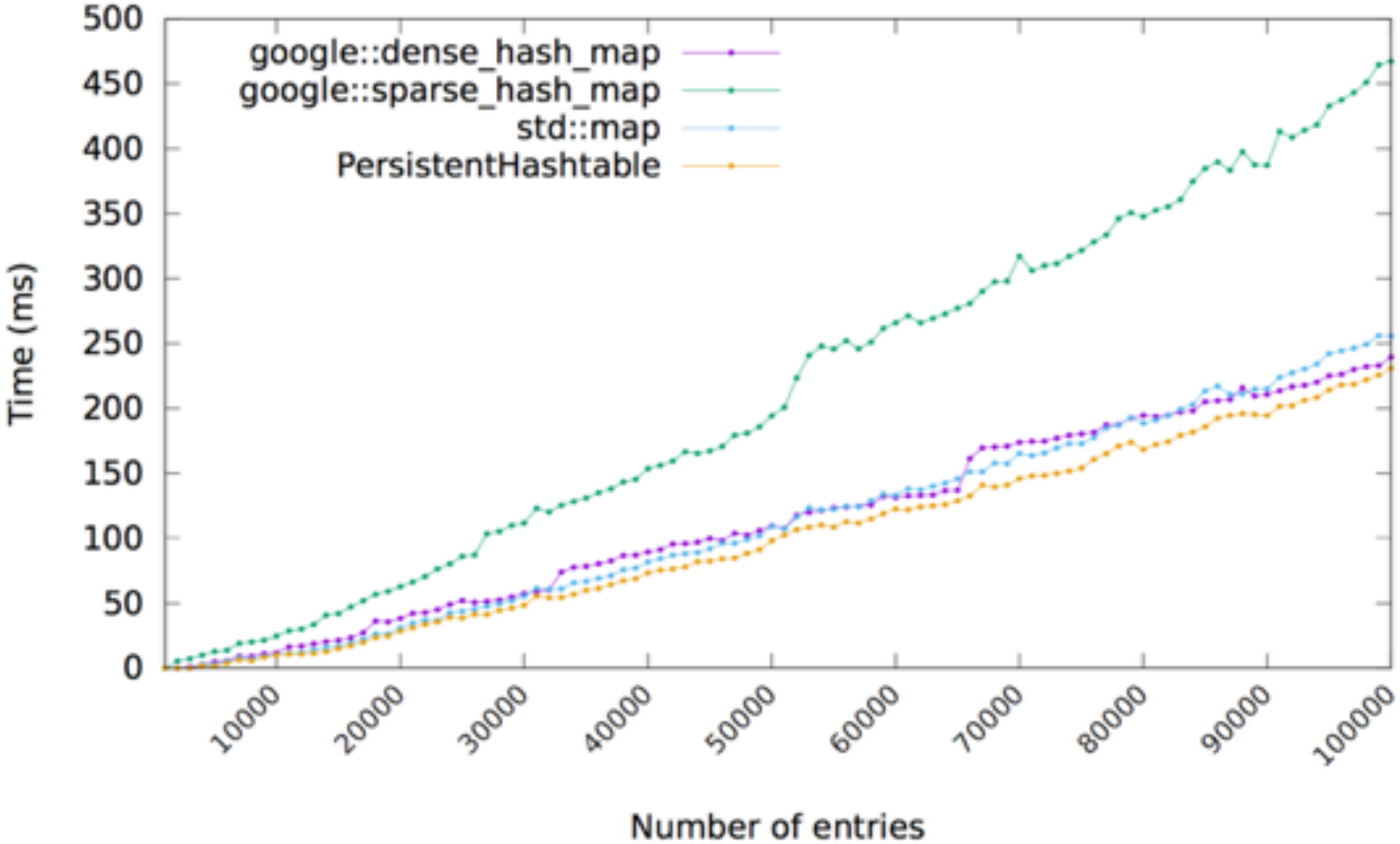
```
pstatic int *p_ptr;

int main (int argc, char const *argv[]) {
    if (p_ptr == NULL) {
        atomic {
            p_ptr = (int*)pmalloc(sizeof(int));
            *p_ptr = 0;
        }
    } else {
        atomic { *p_ptr += 1; }
    }
    printf("*p_ptr = %d\n", *p_ptr );
    return 0;
}
```

Mnemosyne allows easily persistency of C structures

- as first test tried to replace C++ hash with performant C hash
- summer student project for native C hash

PersistentHash

Hash Insertion Performance



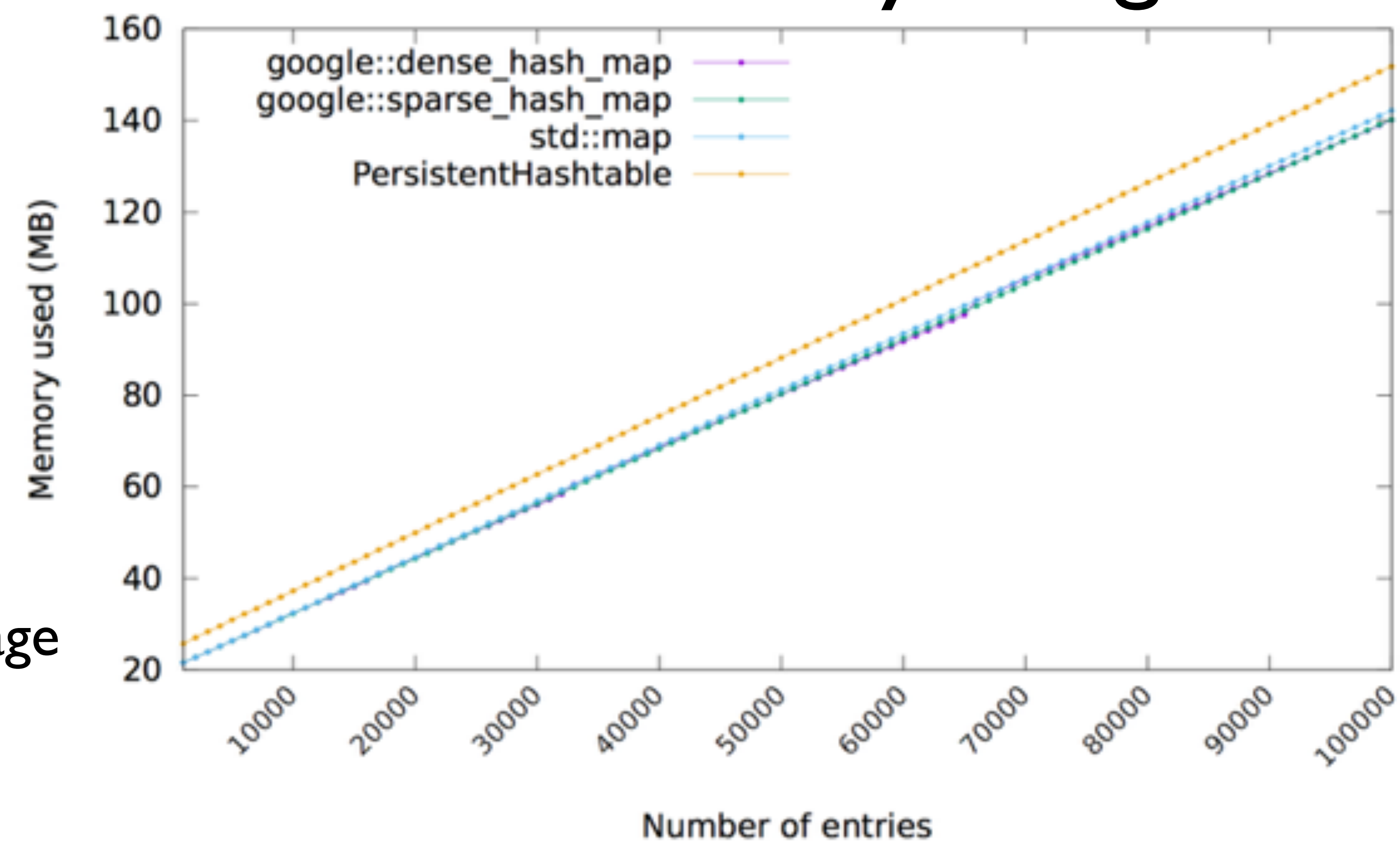
PersistentHash

Benchmark

Implementation and Testing by Marti Bosch

… C can beat C++

Hash Memory Usage



… but slightly more memory usage

Mnemosyne - Evalution

- Mnemosyne needs upgrade to newer gcc/ICC to be able to compile the full EOS code
  - still using old patched gcc compiler
- we need transactional data structures, a hash is a starting point but not sufficient, for e.g.:
  - std::string
  - std::vector
- persistent objects could change the way of thinking while programming
- however the technology is not yet ready
  - for the moment a pure R&D activity without production applicability

Summary

- **Open Kinetic** is a very promising platform to a new way of cloud storage implementation
  - Seagate Kinetic managed to evolve a single vendor solution into an OpenStandard backed by major players on the storage market
  - we did a first demonstration of good scalability and applicability in physics production and analysis workflows
  - Open Kinetic allows to easily scale-out the data volume related part of the EOS storage system by two order of magnitudes
    - the meta-data equivalent technology is currently in development to achieve a similiar scale-out behaviour for meta-data
- **NVRAM** technology is still at the beginning for generic applications and will be interesting once there is a complete support by state-of-the art compilers

CERN