Native SSD Cache for Lustre OSTs

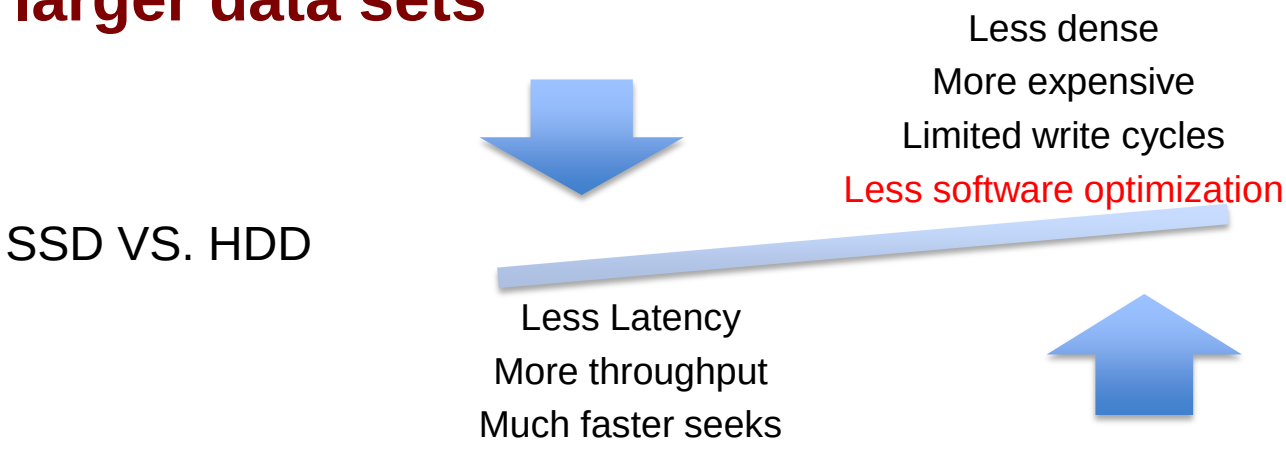
DataDirect Networks, Inc.

Li Xi

2015/10/20

Why SSD cache for Lustre?

- **SSD has better performance than HDD but not always affordable for massive usage**
- **Solution:**
  - Cache + Access Locality = Less cost + Better performance
- **Memory cache sometimes helps but the space is very limited**
- **SSD cache is able to accelerate applications with larger data sets**

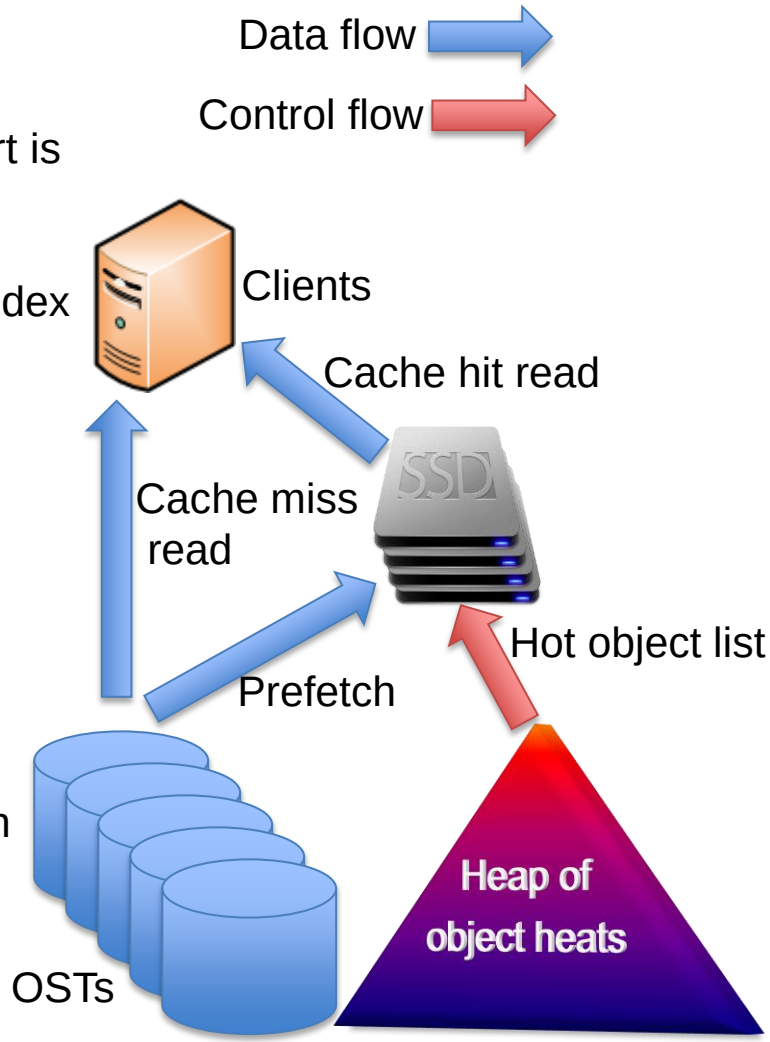


Why implement SSD cache on OSS?

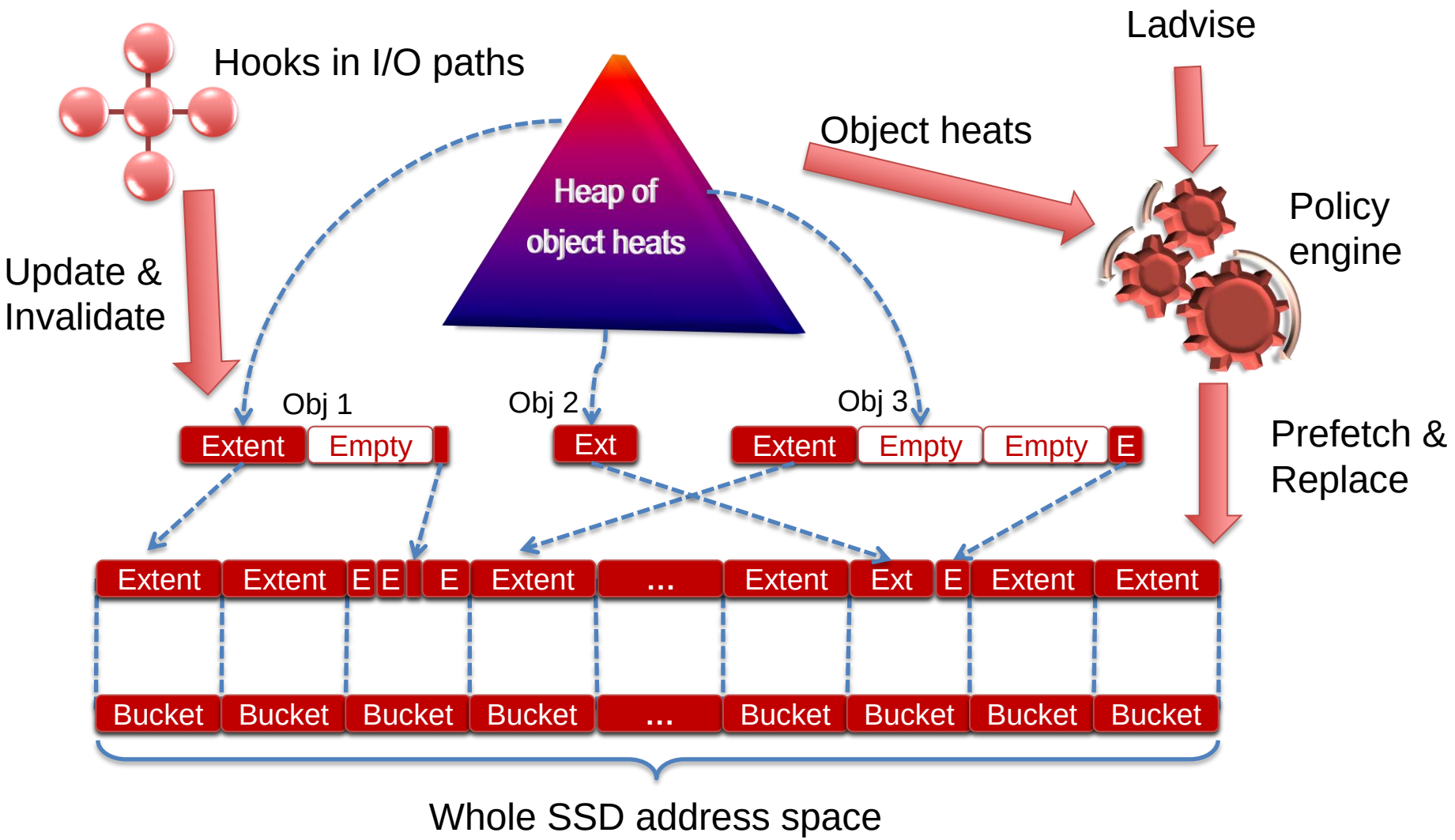
- **SSD cache is not necessary for MDS, since SSDs can be used as storage of MDTs.**
- **SSD on OST devices**
  - Some hybrid drives lack APIs for cache management from Lustre side
  - Cache management APIs are usually different between vendors
- **OST pool of SSDs**
  - Transparent data migration and space management between pools is difficult
- **SSD cache on OSS**
  - We implemented a cache system on OSS named **L2RC（Lustre Level 2 Read Cache）**

Design of L2RC

- **Currently focus on read-only Cache**
  - Read support is relatively easy
  - Write support will be added after read cache support is finished: **L2RC -> L2C**
- **All indexes are kept in memory**
  - Worst-case: 1TB SSD will cost 10GB memory for index
- **Space allocation**
  - All spaces are divided into 1M bucket
  - Bucket is the unit of space management
  - A bucket contains multiple extents
  - Extents size is between 4096 and 1M
  - Cached data of an object is devided to multiple 1M extents and one variable-length extent
- **Prefetch and replace**
  - Automatic management based on **file heat**
  - Cache management by applications/users based on **ladvise**
  - **Hooks** are added in IO path to refresh cache



Space management of L2RC



Design of file heat

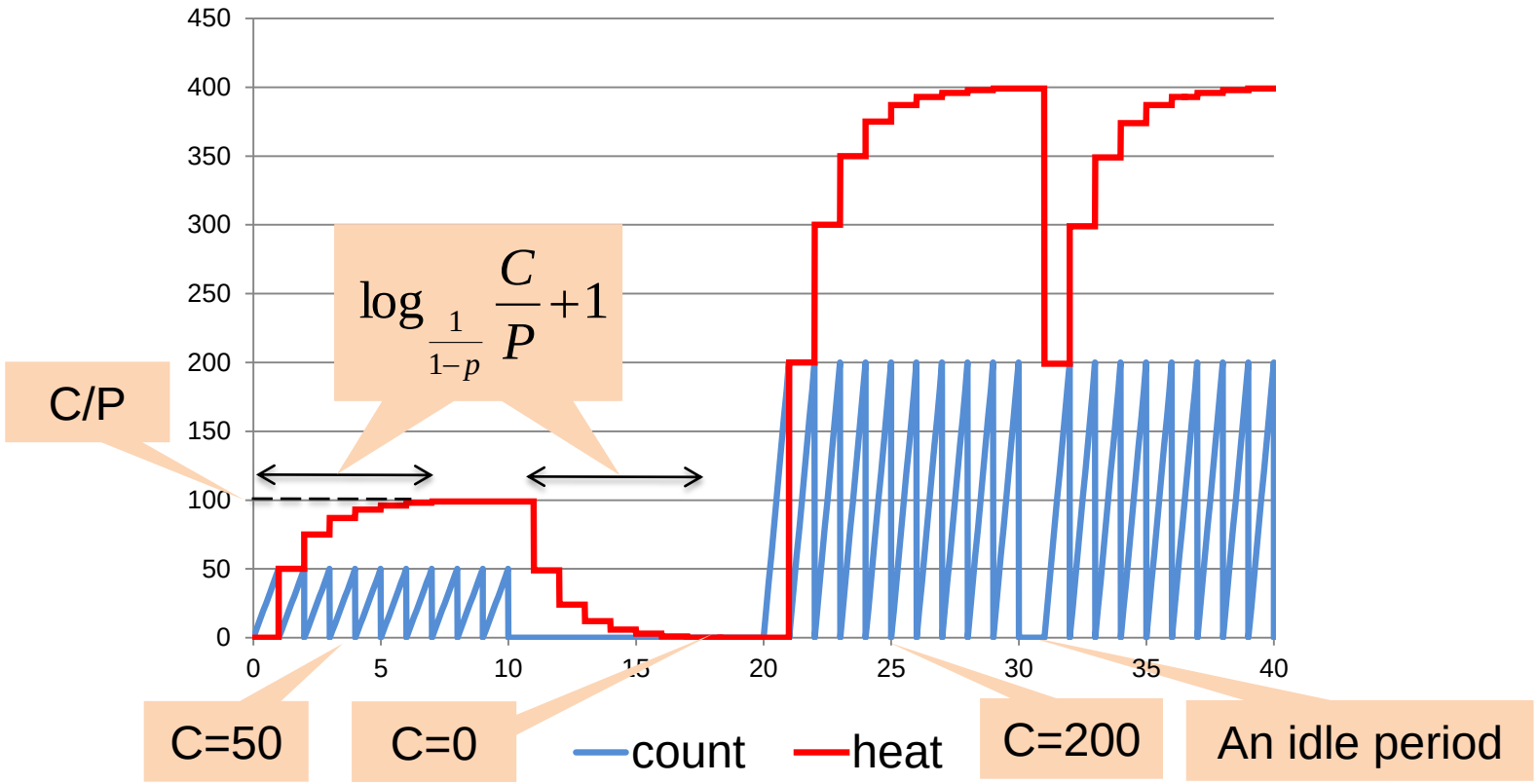
- **Lustre file heat is a relative attribute which can reflect the access frequency of the file/object**
  - It grows as file is accessed, yet dissipates as time moves on
- **Time is divided into periods**
  - The access number during each **period** is counted
  - The file heat is only recalculated at the end of a time period
  - At the end of each time period, a **percentage** of the former file heat is lost

Lower heat
Colder
Less access frequency

Higher heat
Hotter
More access frequency

What does file heat look like?

C: Access count per time period
P: Loss percentage per time period



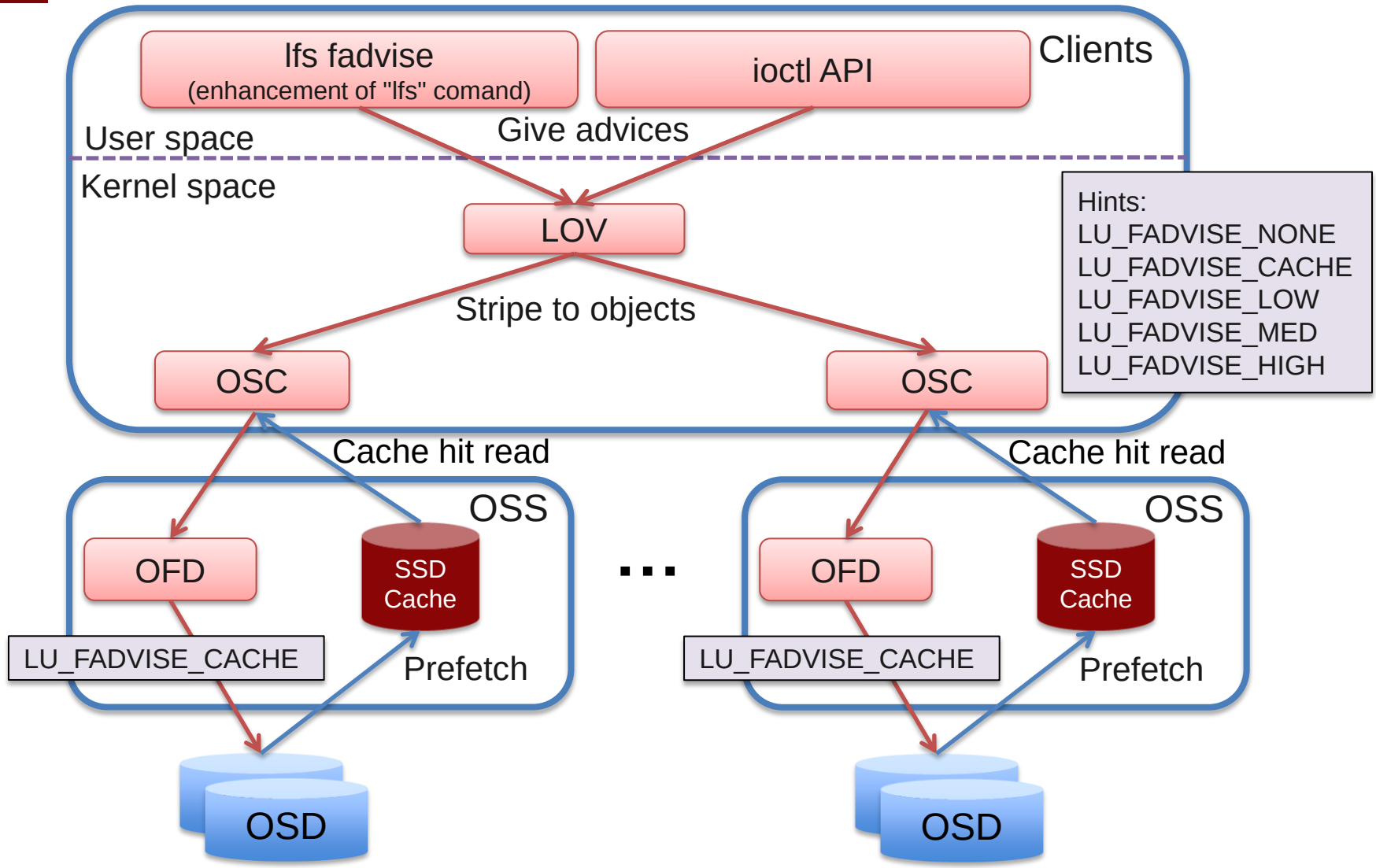
Why file heat for L2RC?

- **Local cache algorithms such as LRU is not suitable for SSD cache**
  - They are not able to sort distributed objects
  - They are designed for RAM/CPU, but prefetch of SSD cache costs much more than RAM/CPU
- **Multiple heat instances are supported**
  - read_samples, write_samples, read_bytes, write_bytes, metadata_updates, etc.
- **Heat instances can be synthesized to an aggregative indicator**
  - High read heat + Low write heat = Good to prefetch to read-only cache
  - High read heat + High write heat = Good to prefetch to read&write cache, not good to prefetch to read-only cache
  - Low read heat + Low write heat = Good to archive to backup system

Prefetch based on ladvise

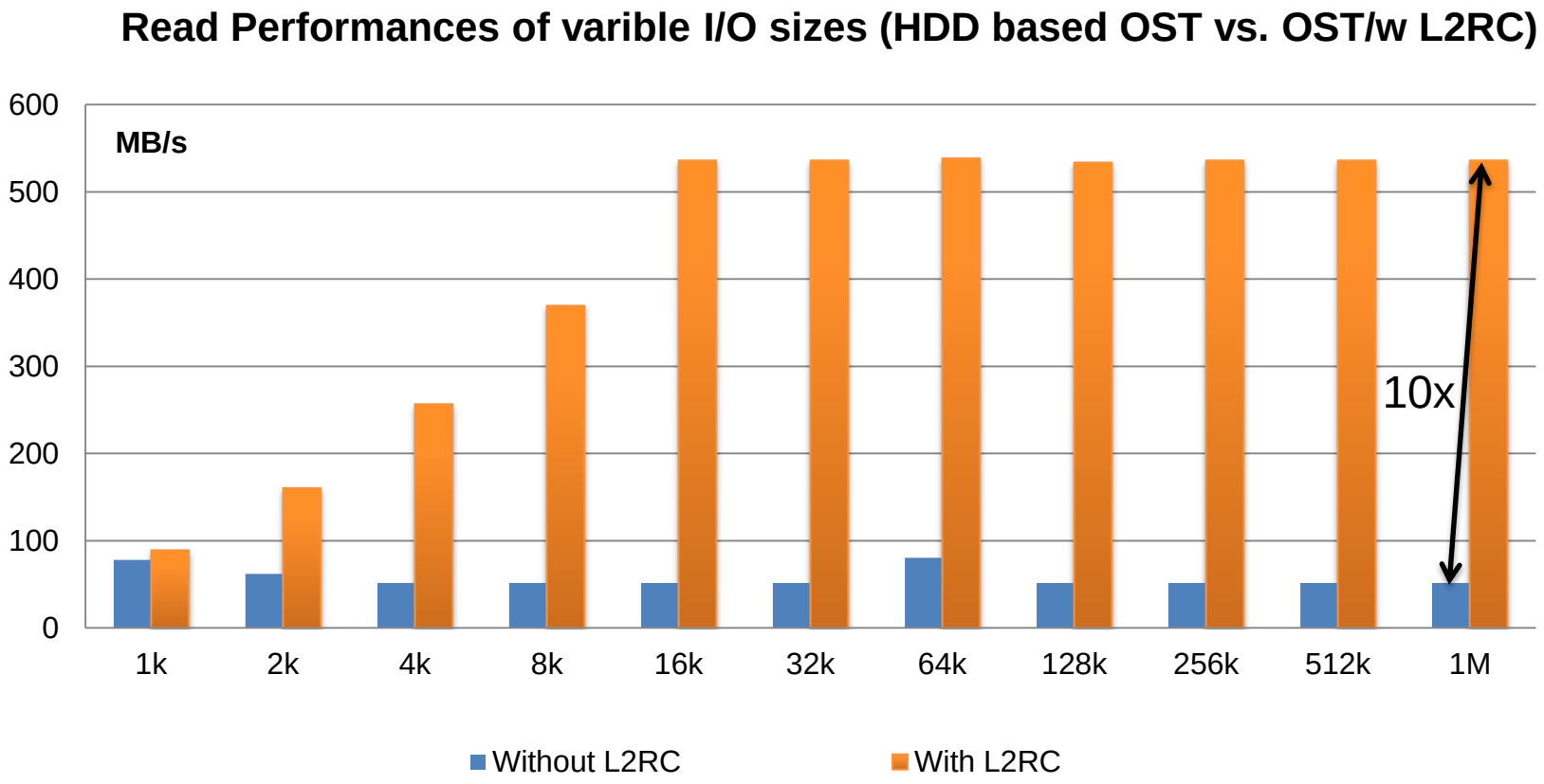
- **Ladvise**
  - A framework of Lustre to send advices/hints from clients to server components.
- **Ladvise is integrated into Lustre stacks**
  - Transparently handle file stripe of Lustre
  - Give hints through the I/O path to keep efficient
- **Utility and API is simple to use**
  - “lfs fadvise” command to give advices on Lustre files
  - ioctl(LL_IOC_FADVISE) for advices from smart applications
- **Ladvise enables applications and users with external knowledge to intervene in cache management**

Architecture of ladvise



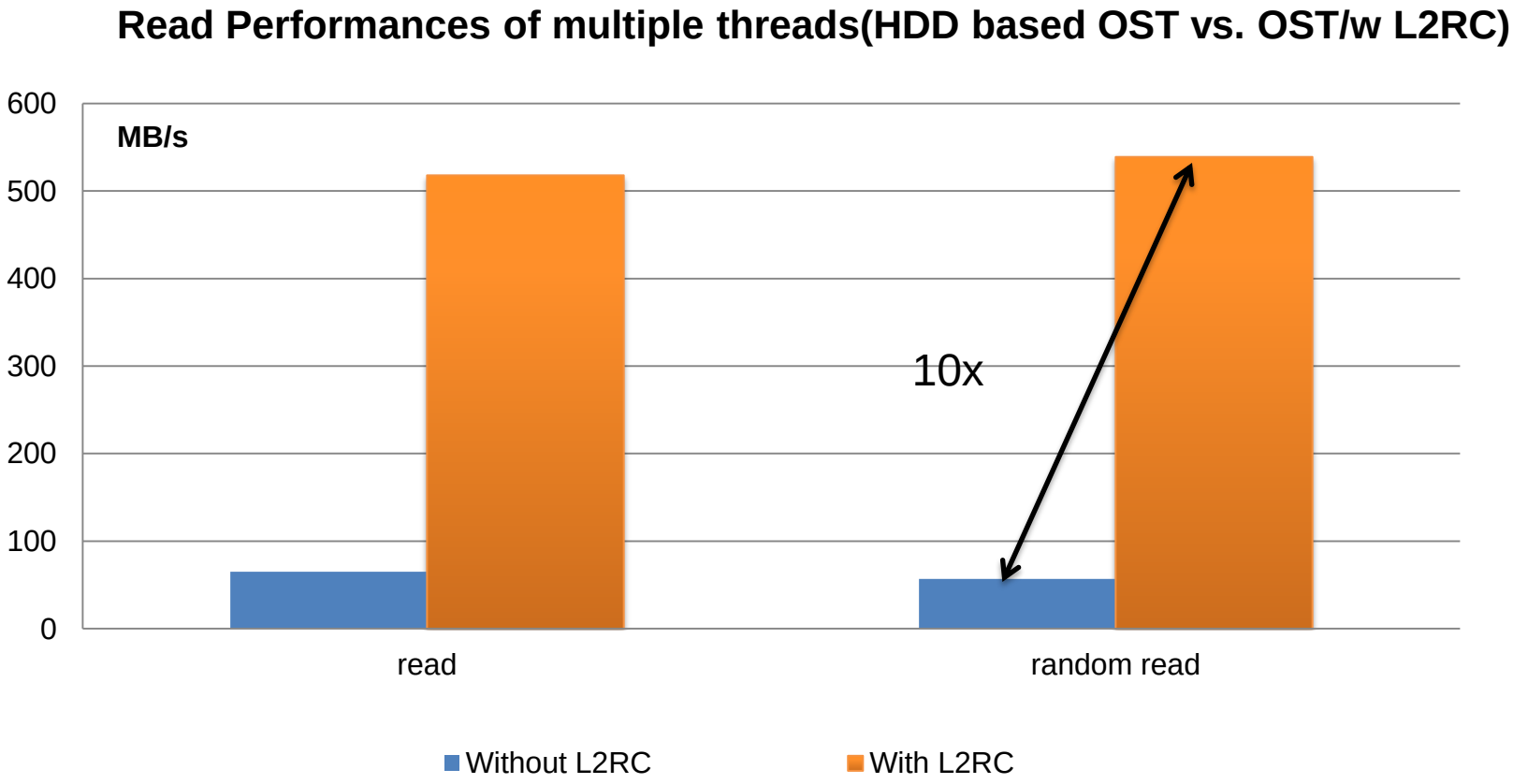
Benchmark results 1

- **One big file of 120GB, single thread (dd)**



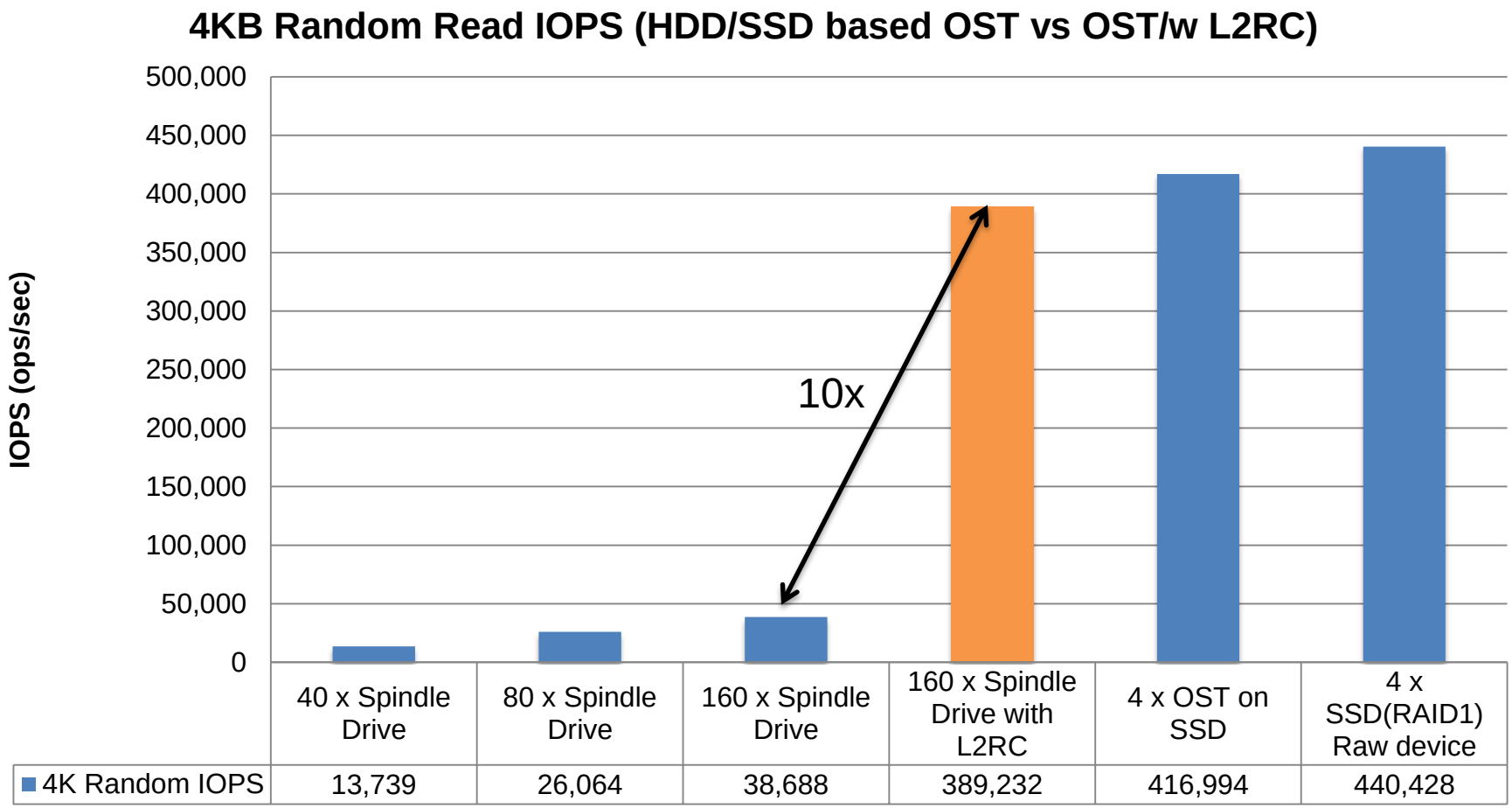
Benchmark results 2

- **Multiple thread performance (tiobench with 4 threads)**



Benchmark results 3

- **IOPS of multiple clients (IOR with 32 clients)**



Conclusion

- We implemented an SSD cache named L2RC on Lustre OSS
- We provided automatic cache management mechanism based on file heat, as well as APIs based on ladvise
- We proved that L2RC is able to present the maximum performance of SSD to applications
- We demonstrated that L2RC might be able to accelerate read performance of different applications

Thank you!