슈퍼컴퓨팅과 데이터 어낼리틱스 프레임워크, 그리고 유니스트의 빅데이터 처리 프레임워크

DEVIEW
2015

남범석
UNIST (울산과학기술원)

contents

DEVIEW
2015

1. 슈퍼컴퓨팅과 빅데이터
2. UNIST의 빅데이터 분석 플랫폼
3. 맺음말

1. Supercomputing & BigData

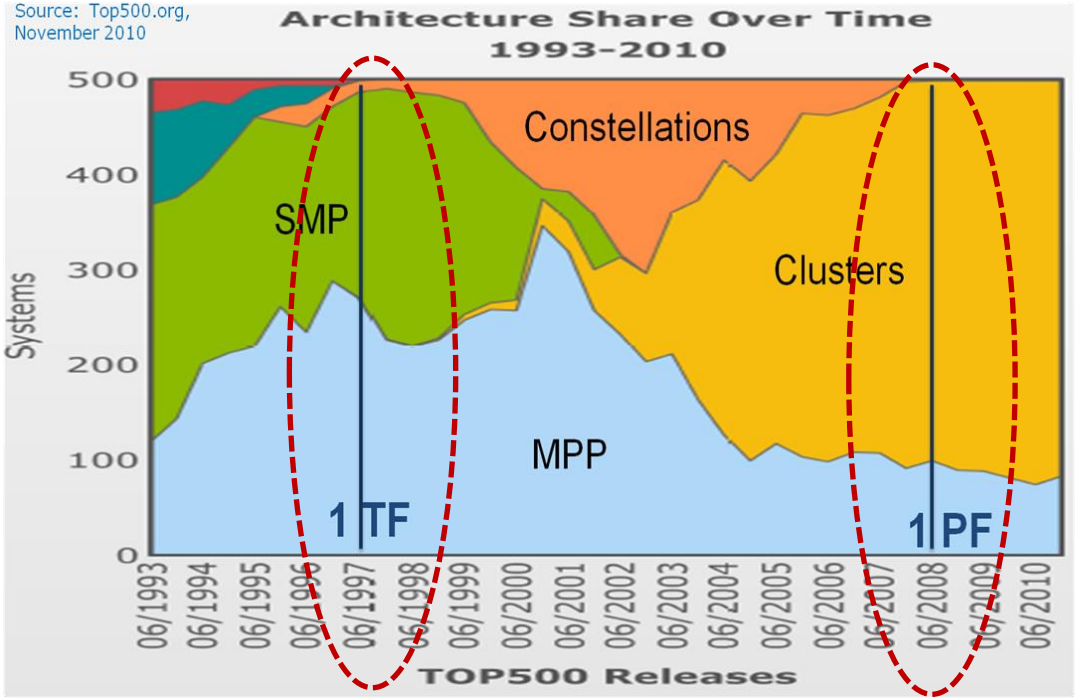
Supercomputing

- High Performance Computing
- Scientific Computing
- Data Intensive Computing

- Computational Science /Rocket Science
- High-end Computing

Supercomputing 아키텍처

1980's 1990's 2000's



Source: Top 500

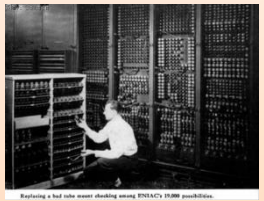
Supercomputing Today

Hardware ecosystem

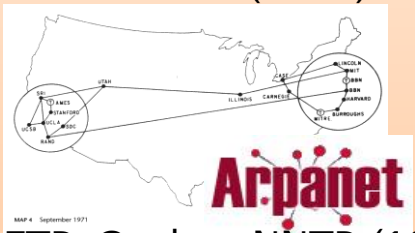
- Clusters of commodity (Intel/AMD x86) HW & IBM bluegene
- High-speed, low-latency interconnect (Infiniband)
- Coprocessors (GPU, Xeon Phi)
- Storage area networks (SANs) & local disks for temporary files

High-end Computing & Industry

HPC



ENIAC (1946)

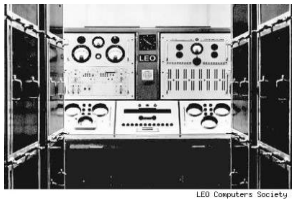


FTP, Gopher, NNTP (1969)

CERN

CERN 통신 시스템(1989)

NCSA Mosaic (1993)

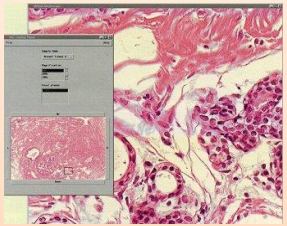


최초 Business Computer (1951)

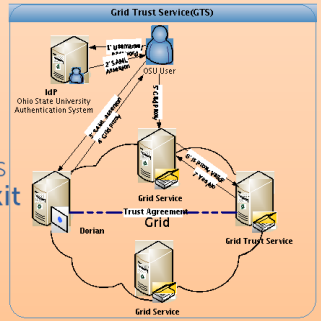
World Wide Web
dot-com bubble (1997~)

High-end Computing & Industry

HPC

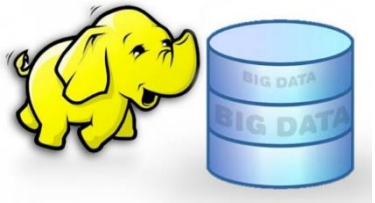


Active Data Repository (1998)



globus toolkit

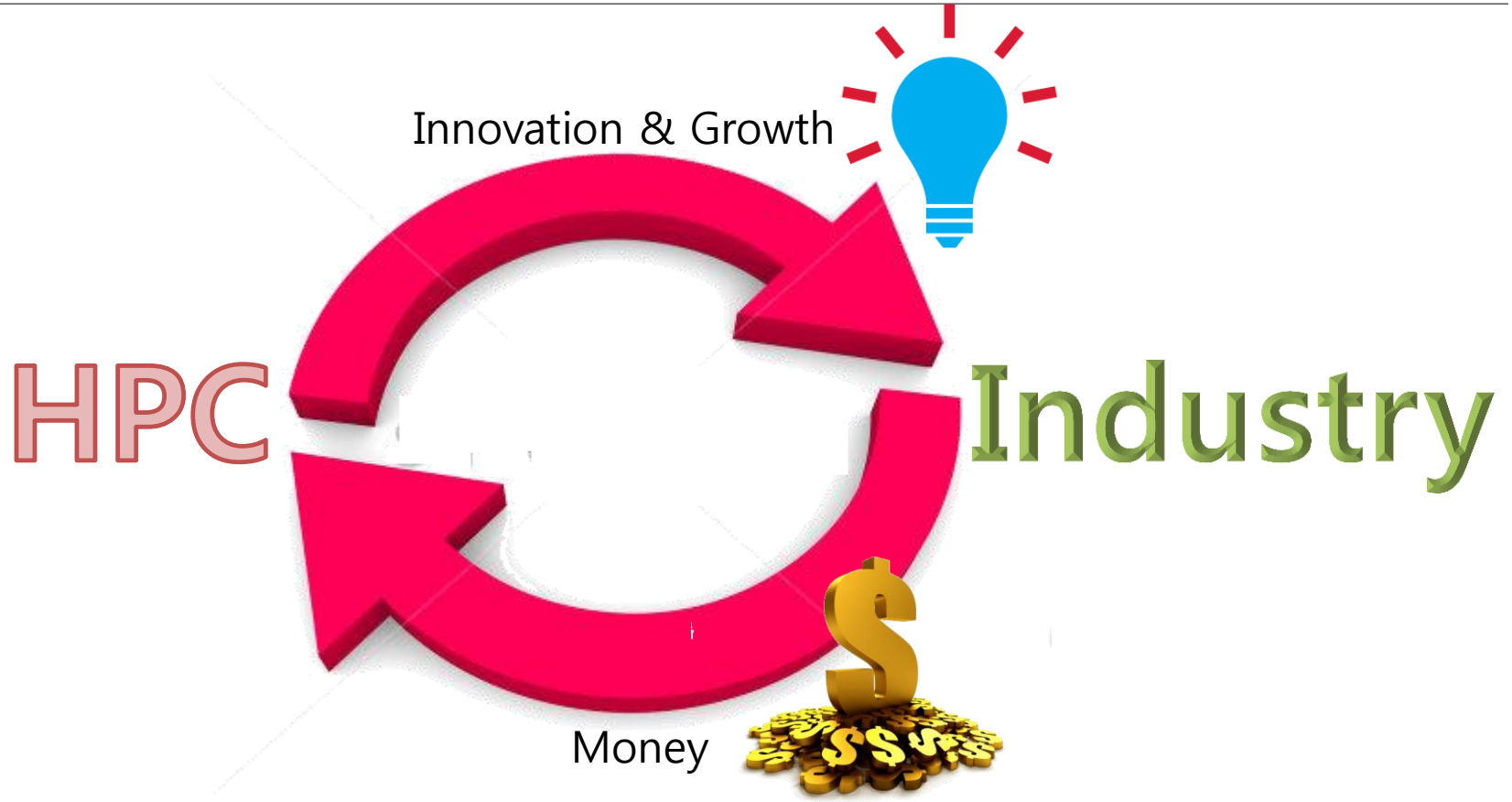
Grid Computing (1998)



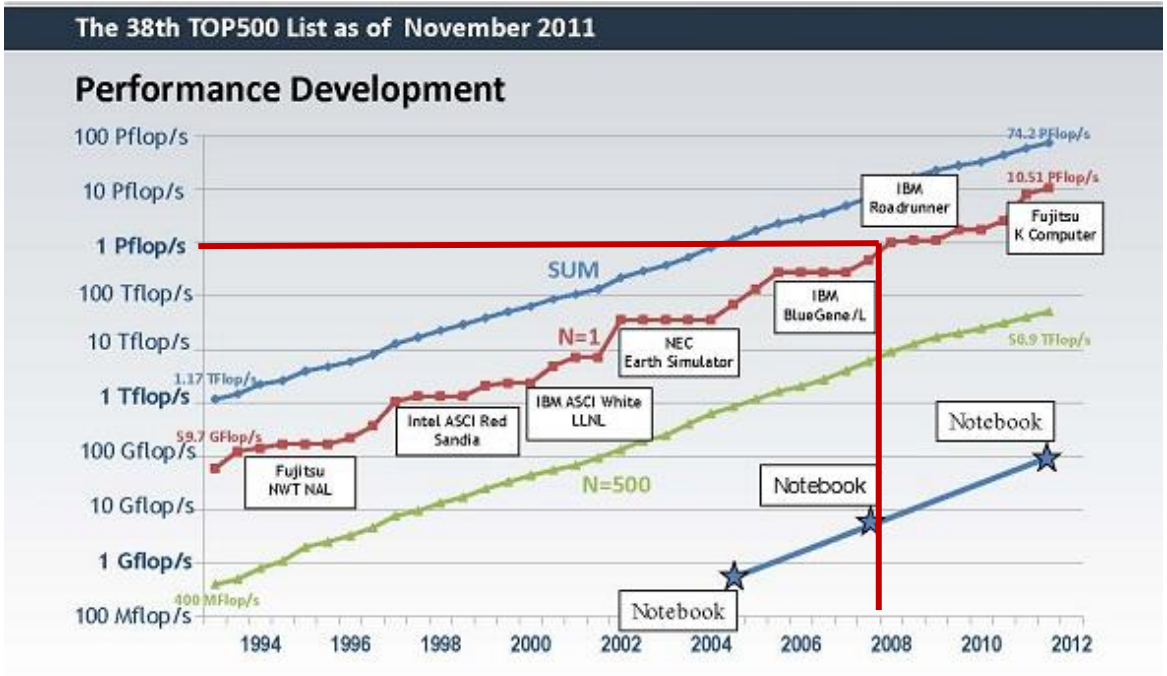
BigData/Hadoop (2005)

Amazon EC2 (2006)

HPC & Industry

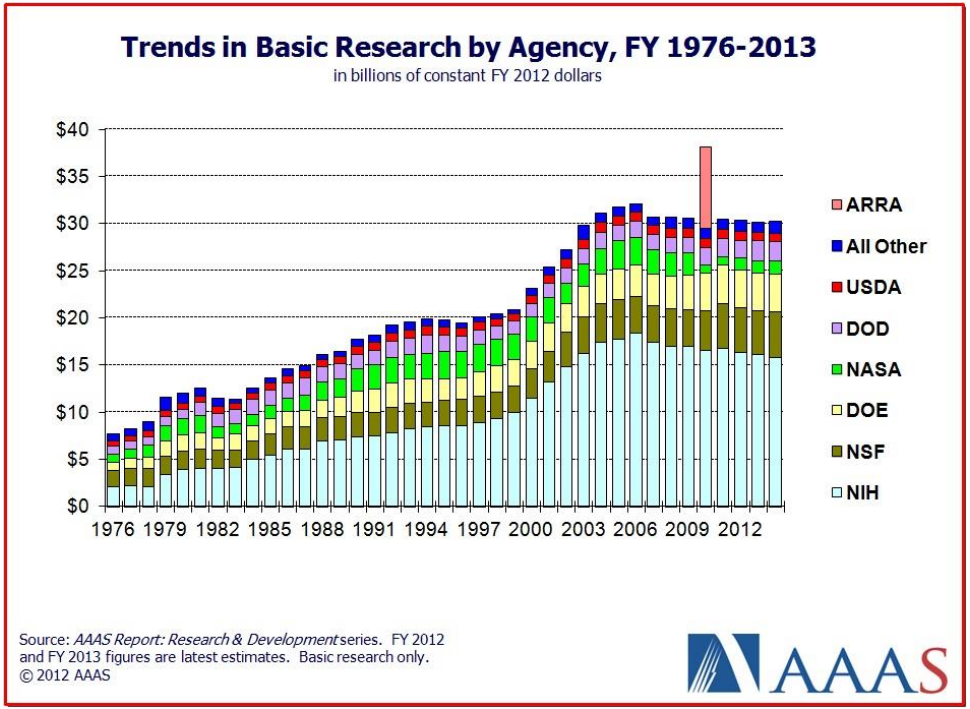


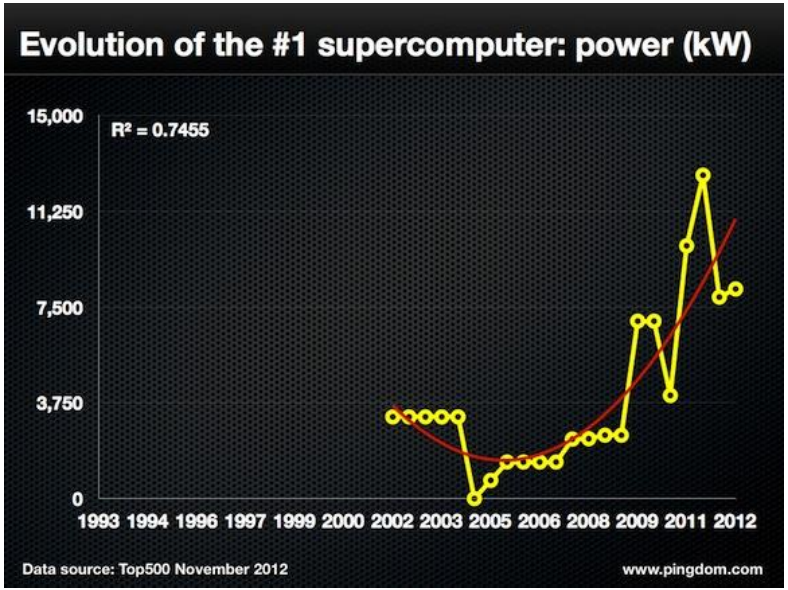
2008



Source: Top 500

Recession & HPC





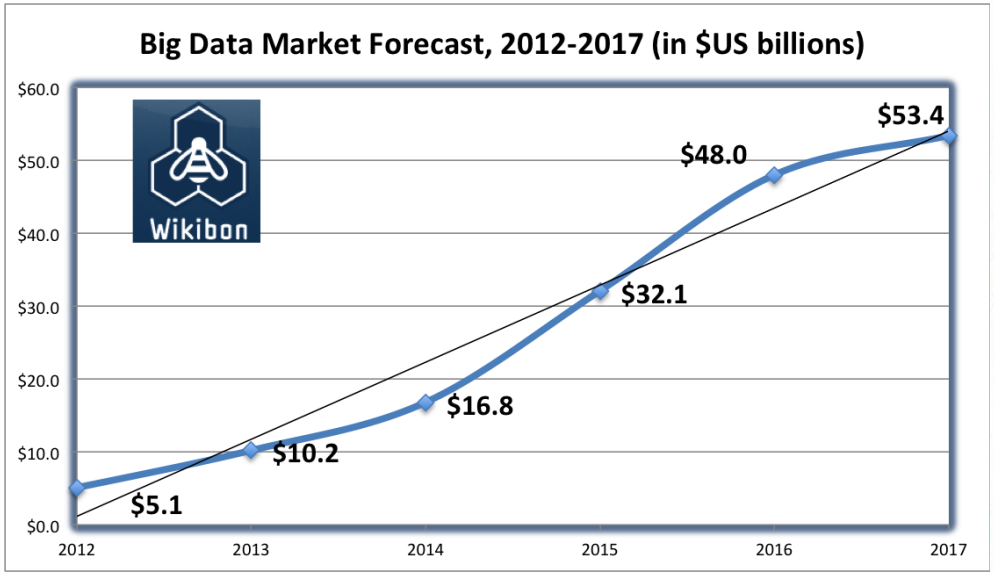
Exa 스케일? 연간 수천만달러 전기세

Source: AAAS, Top 500

BigData: Recession?

DEVIEW
2015

2005



Source: Gartner, Apache Foundation

BigData? HPC?

Isn't that what I've done in the past?

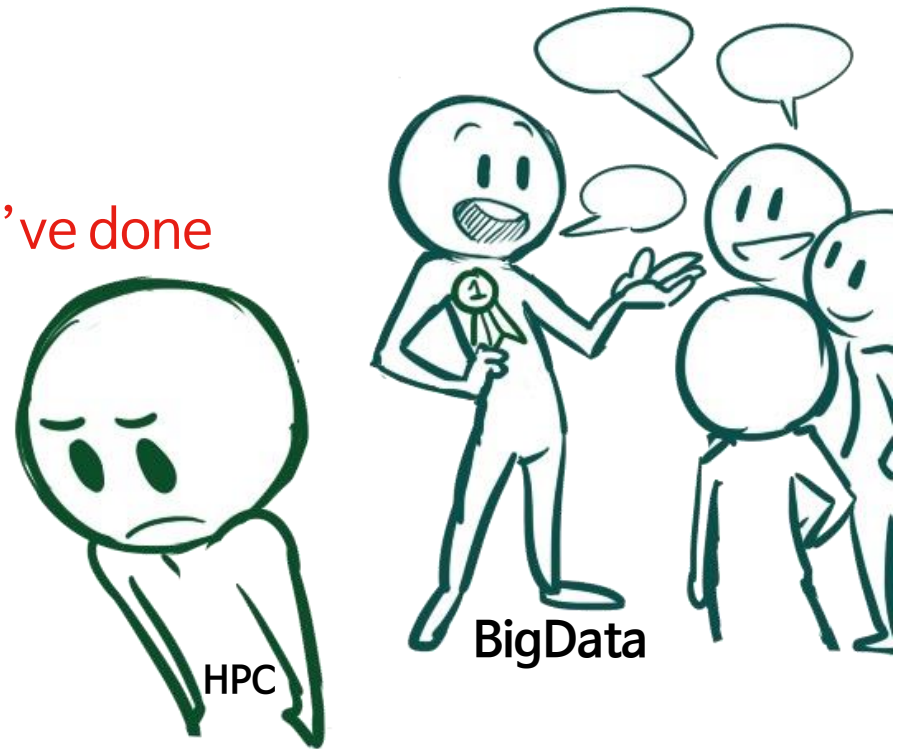


Image Source: solar-citrus.tumblr.com

Hadoop's Uncomfortable Fit in HPC

Why?

1. Hadoop is an invader!
2. Running Java applications (Hadoop) on supercomputer looks funny
3. Hadoop reinvents HPC technologies poorly
4. HDFS is very slow and very obtuse

Rob Meyer
@rmeyernag

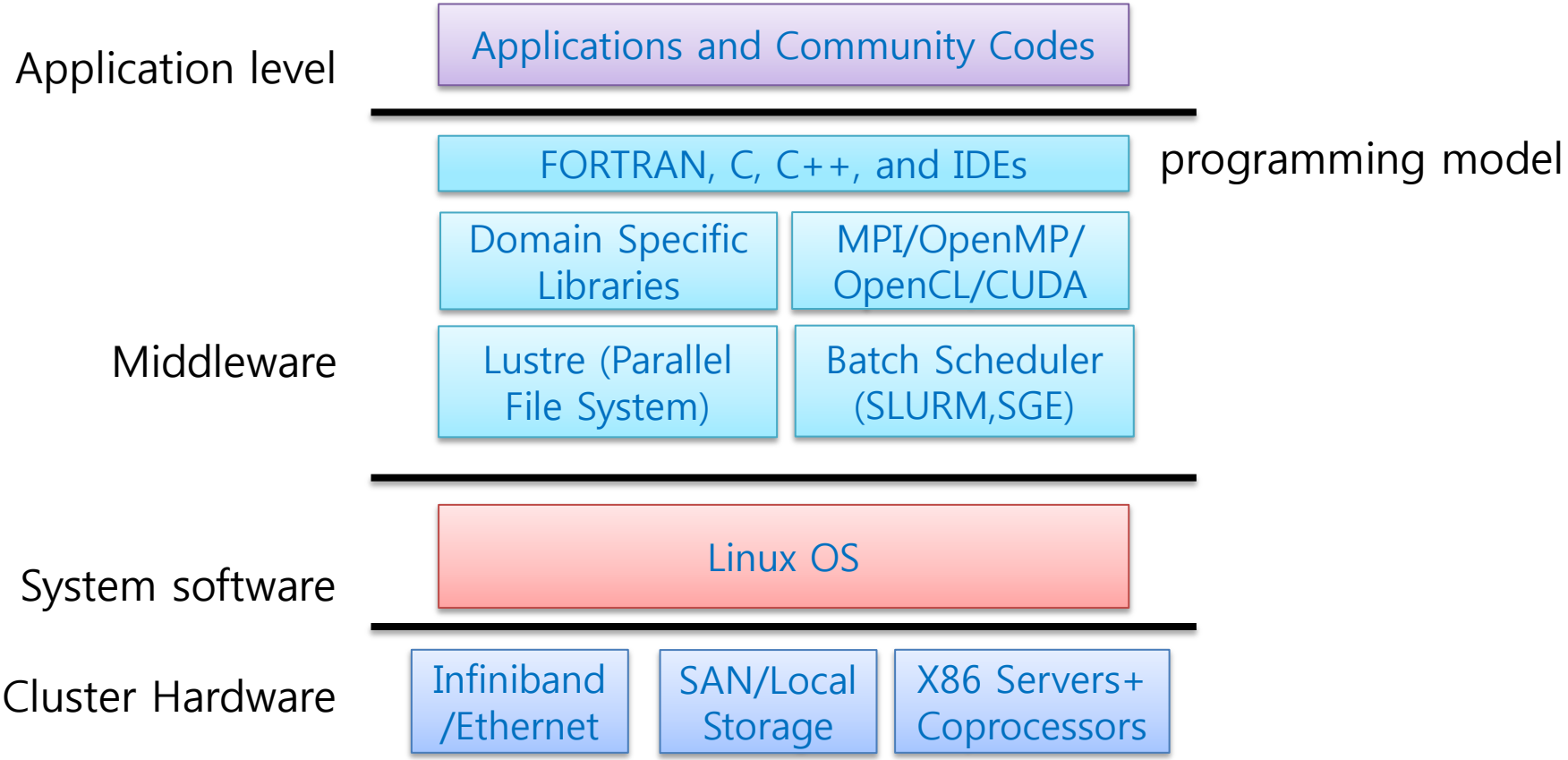
Spent the entire day @ncsa listening to smart people talk about big data and no one person mentioned Hadoop. How weird is that?

5/14/14, 1:25 PM

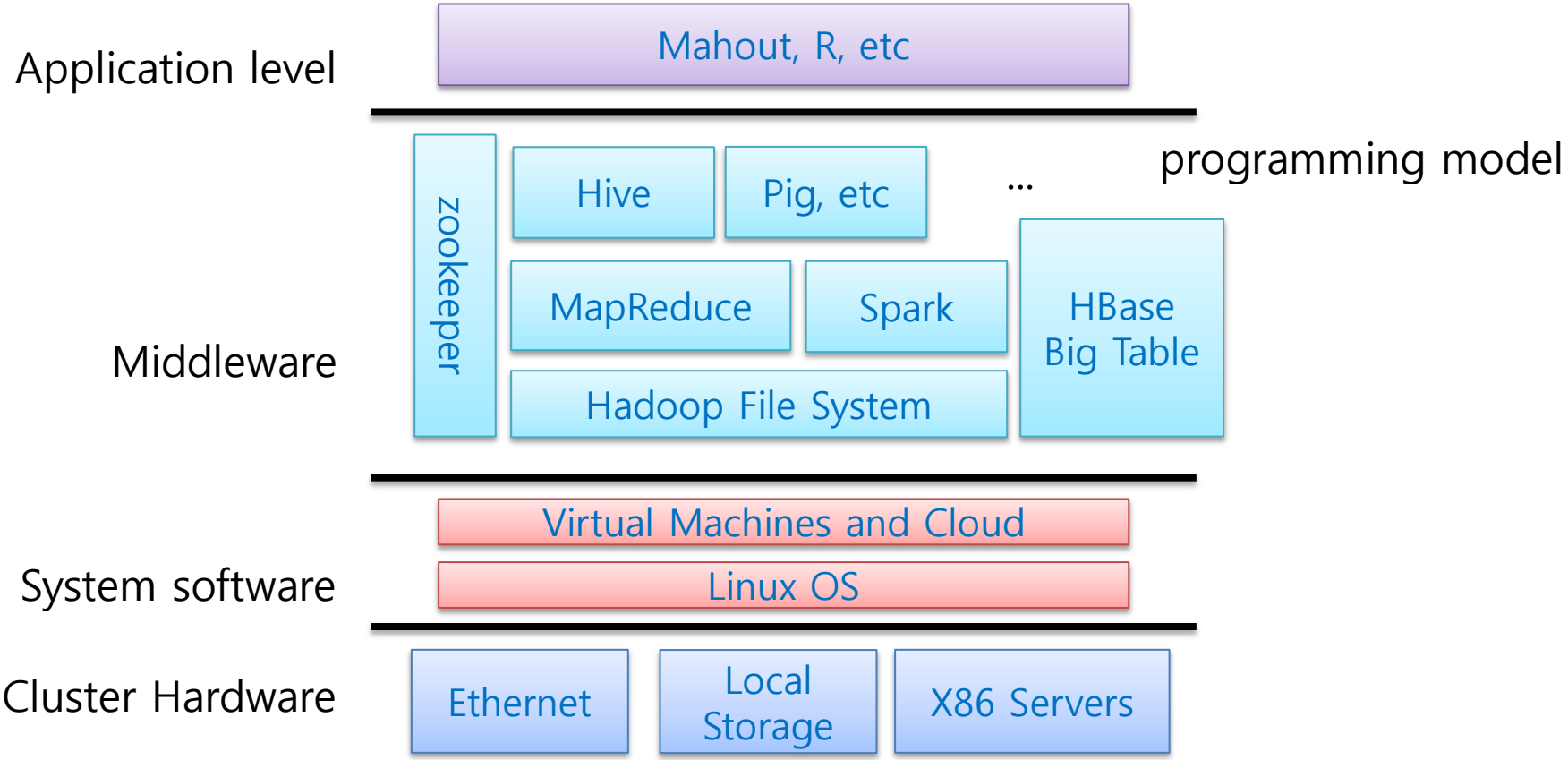
Source: G. K. Lockwood, HPCwire

Image Source: solar-citrus.tumblr.com

HPC Ecosystem



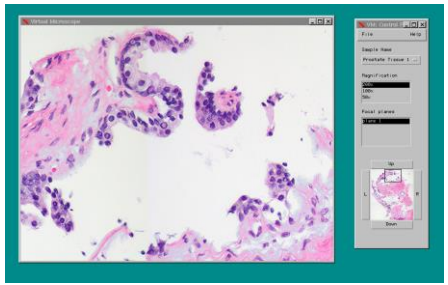
BigData Ecosystem

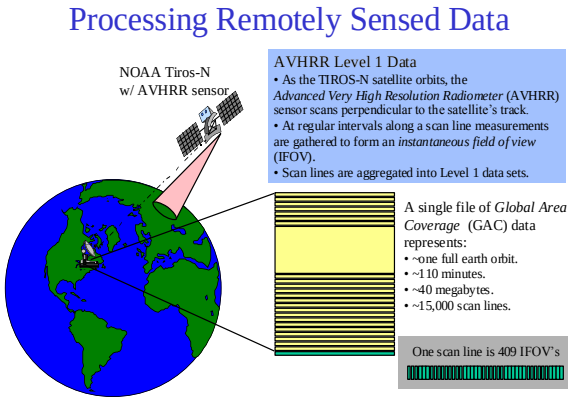


ADR vs MapReduce

1998 Active Data Repository (ADR)

- Data Intensive Computing Framework
- Scientific Datasets 처리





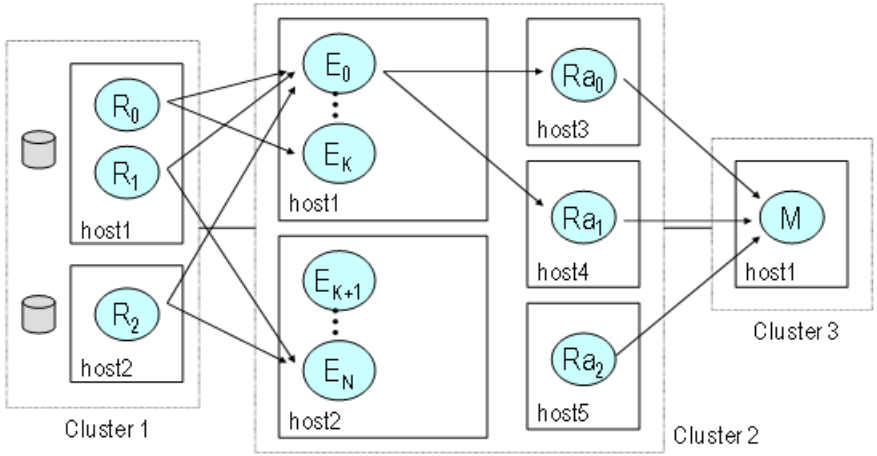

```
O ← Output dataset, I ← Input dataset
A ← Accumulator (intermediate results)
[S_I, S_O] ← Intersect(I, O, R_query)
foreach o_e in S_O do
        read o_e
        a_e ← Initialize(o_e)
foreach i_e in S_I do
        read i_e
        S_A ← Map(i_e) ∩ S_O
        foreach a_e in S_A do
                a_e ← Aggregate(i_e, a_e)
foreach a_e in S_O do
        o_e ← Output(a_e)
        write o_e
```

Source: Chaos project: U.Maryland, College Park

DataCutter vs Dryad, Spark

2000 DataCutter

- Workflow 지원 Component Framework
- Pipelined components (filter)
- Stream based communication




```
class MyFilter : public Filter_Base {
   public:
        int init( int argc, char * argv[] )
        { ... };
        int process( stream_t st[] ) { ... };
        int finalize( void ) { ... };
}
```

Source: Chaos project: U.Maryland, College Park

HPC & BigData

HPC 와 BigData 의 공통 해결 과제

1. High-performance interconnect technologies
2. Energy efficient circuit, power, and cooling technologies
3. Power and Failure-aware resilient scalable system software
4. Advanced memory technologies
5. Data management- volume, velocity, and diversity of data
6. Programming models
7. Scale-up? Scale-out?

HPC for BigData

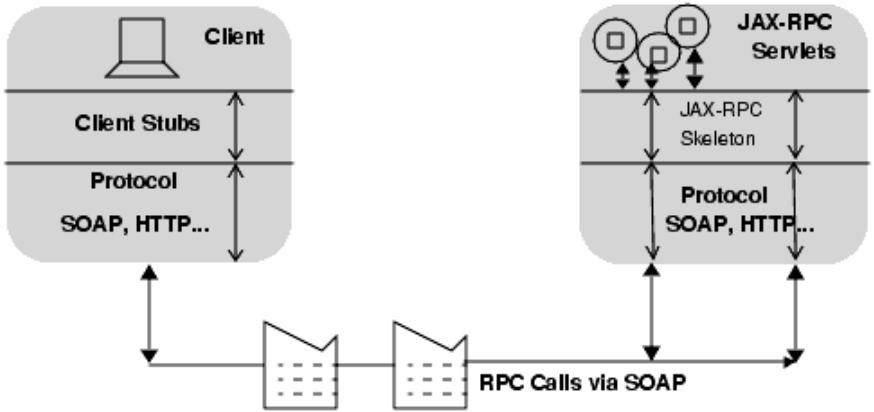
Infiniband/RDMA for Hadoop

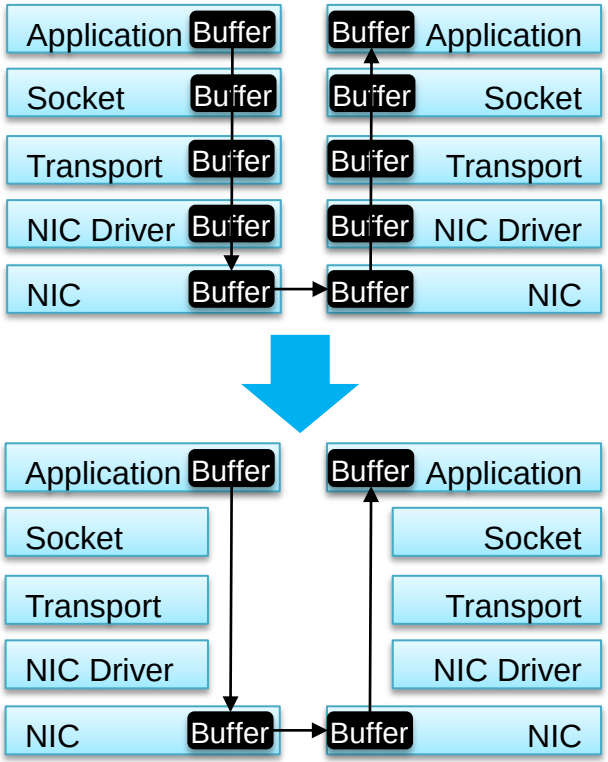
- Wide adoption of RDMA
  - Message Passing Interface (MPI) for HPC
  - Parallel File Systems
    - Lustre
    - GPFS
  - Delivering excellent performance
    - (latency, bandwidth and CPU Utilization)
  - Delivering excellent scalability

HPC for BigData

Infiniband/RDMA for Hadoop

- Socket 기반의 Java RPC
- HPC는 고속 lossless DMA 통신 사용



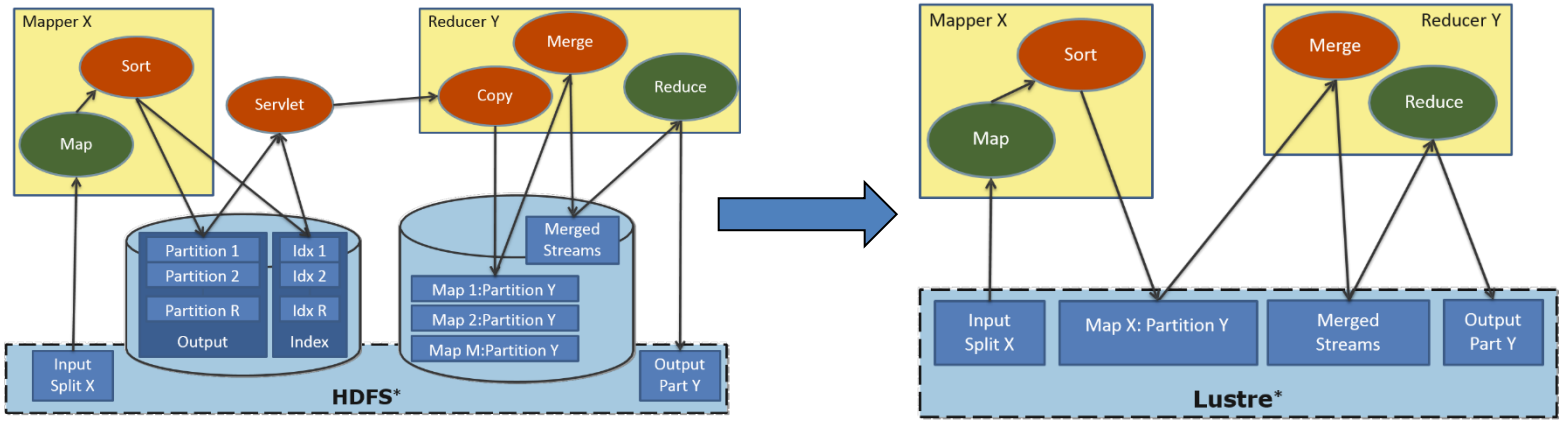


Source: D.K.Panda, ISCA15 tutorial

HPC for BigData

Lustre File System for Hadoop

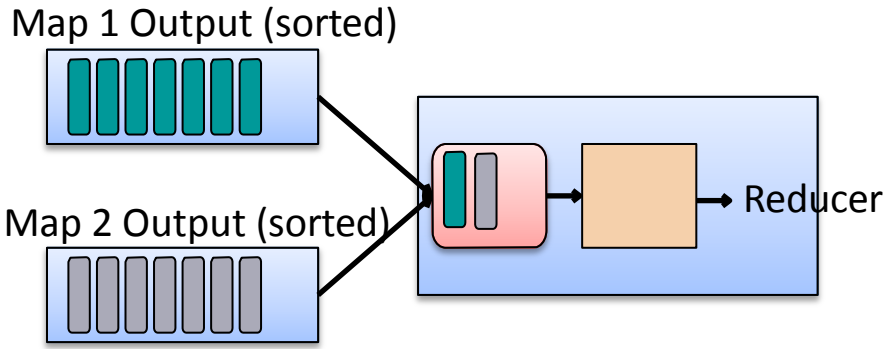
- Intel HPC distribution for Hadoop
  - Map task의 중간 결과물을 Lustre FS 에 저장

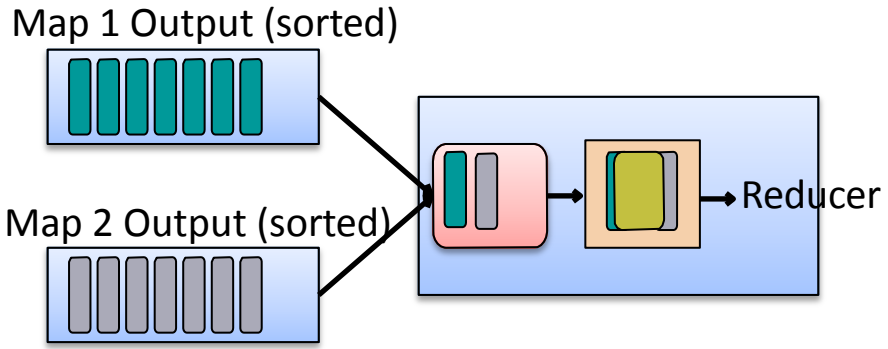


Source: Intel

RDMA 기반 In-Memory Merge

1. Sorted map output is divided into small pieces based on shuffle packet size and size of the key-value pairs
2. As small portion of data is shuffled, merge can take place in memory



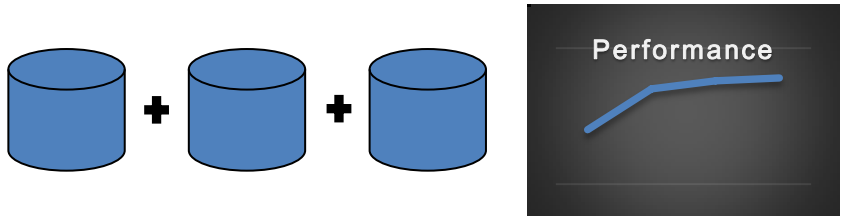


Source: D.K. Panda, ISCA 15 Tutorial

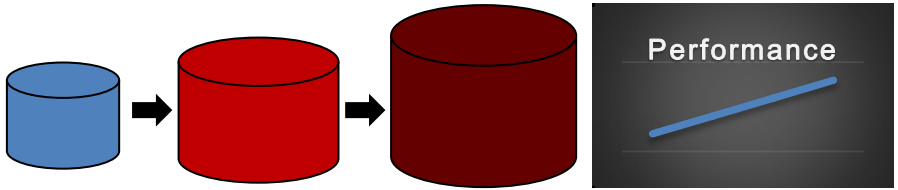
HPC for BigData

Scale out vs Scale up

1. Scale-out



2. Scale-up



Scale-up is more cost-effective
(Microsoft – SOCC 2013)

EclipseMR

DEVIEW
2015

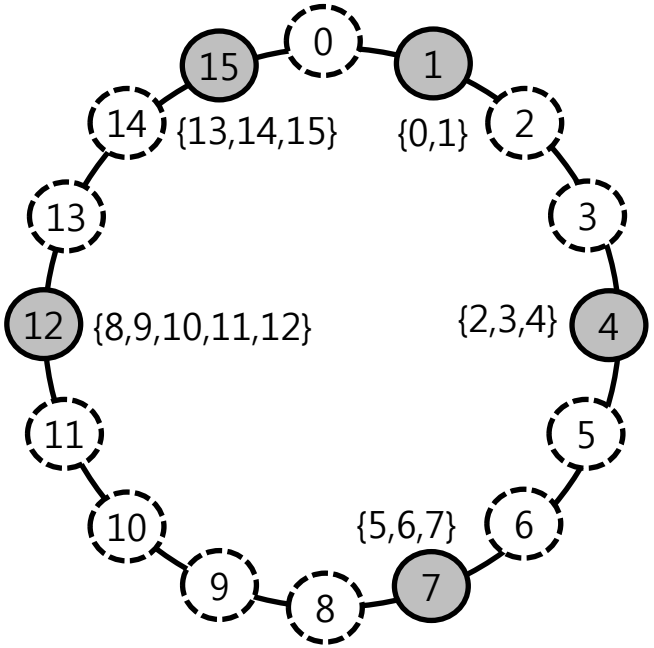
남범석
UNIST (울산과학기술원)

contents

2. EclipseMR

DHT 기반 빅데이터 프레임워크

CHORD DHT (Distributed Hash Table,분산해시테이블)



특징

1. Decentralized Data Structure
2. 결함 내성
3. 확장성 (P2P)
4. 바이너리 서치/Short cut->빠른 데이터 검색

Chord DHT

Consistent Hashing

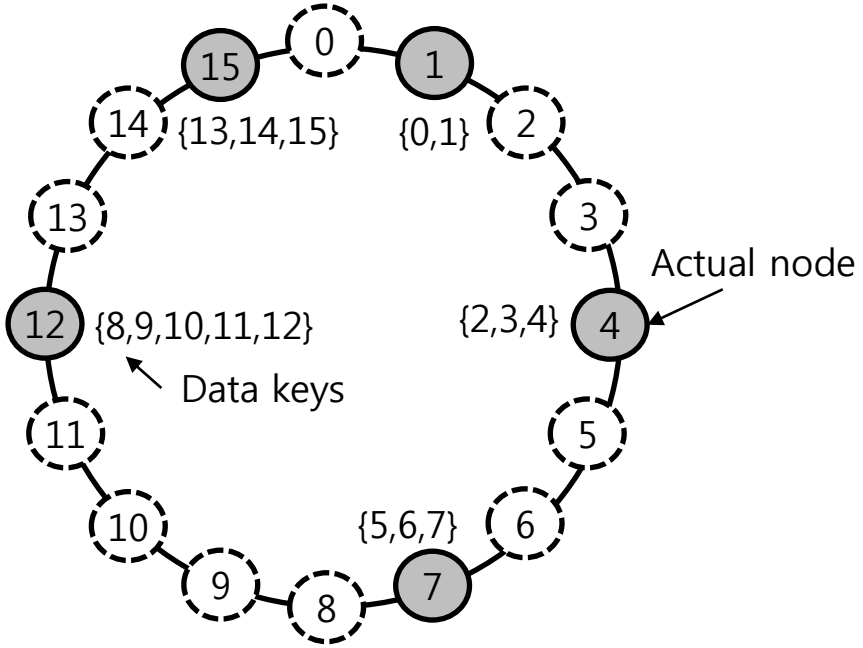
1. Hash function assigns a key to each node and each data
   - key = hash(IP,port)
   - key = hash(dataID)

ex) Insert new_data:

10 = hash(new_data)

12 = successor(10)

: successor(X) : actual node following X



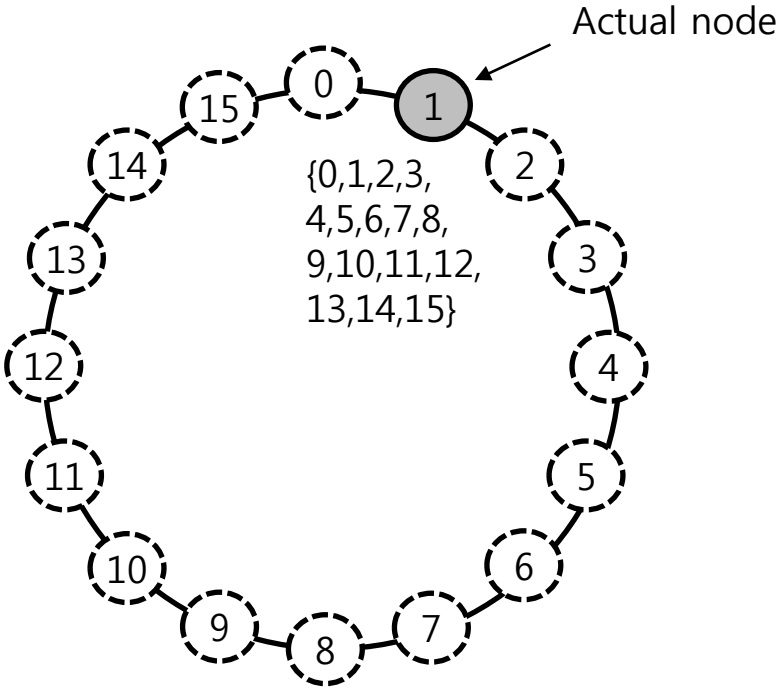
Chord DHT

Join Algorithm

ex) A new server X joins

: 9 = hash(X_IP,X_port)

: 1 = Search(9)



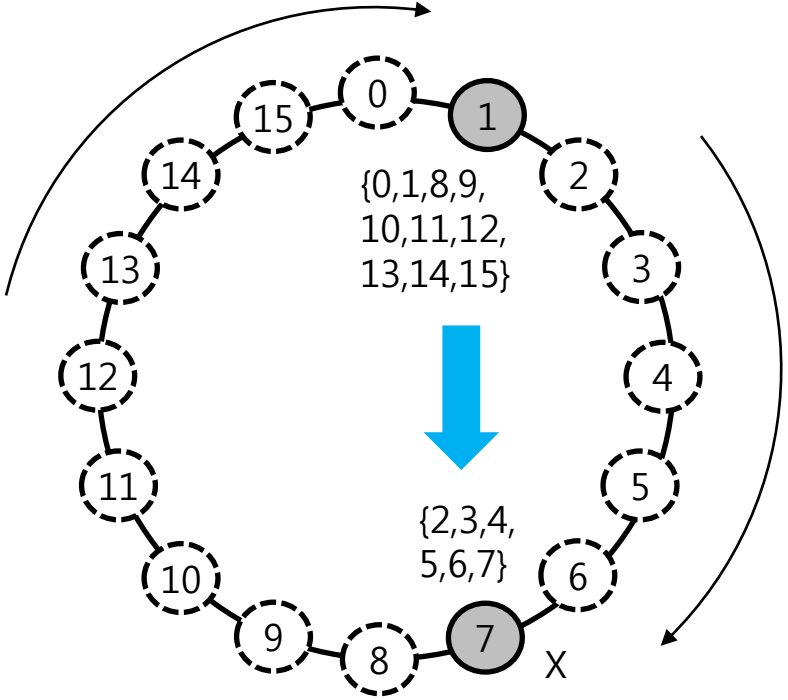
Chord DHT

Join Algorithm

ex) A new server X joins

:7= hash(X_IP,X_port)

: 1 = Search(7)



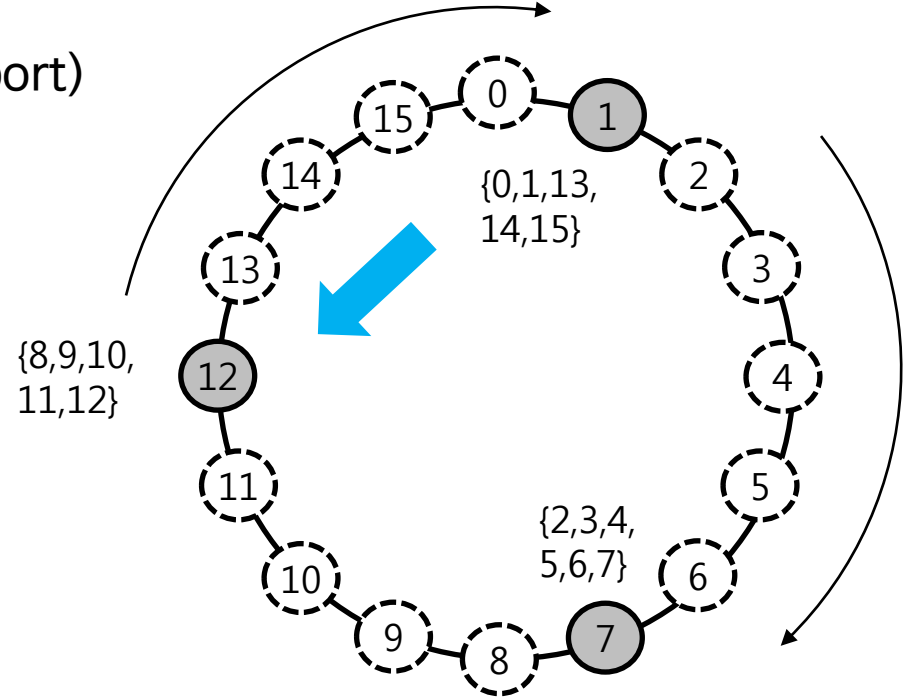
Chord DHT

Join Algorithm

ex) A new server Y joins

: 12 = hash(Y_IP,Y_port)

: 7 = Search(12)



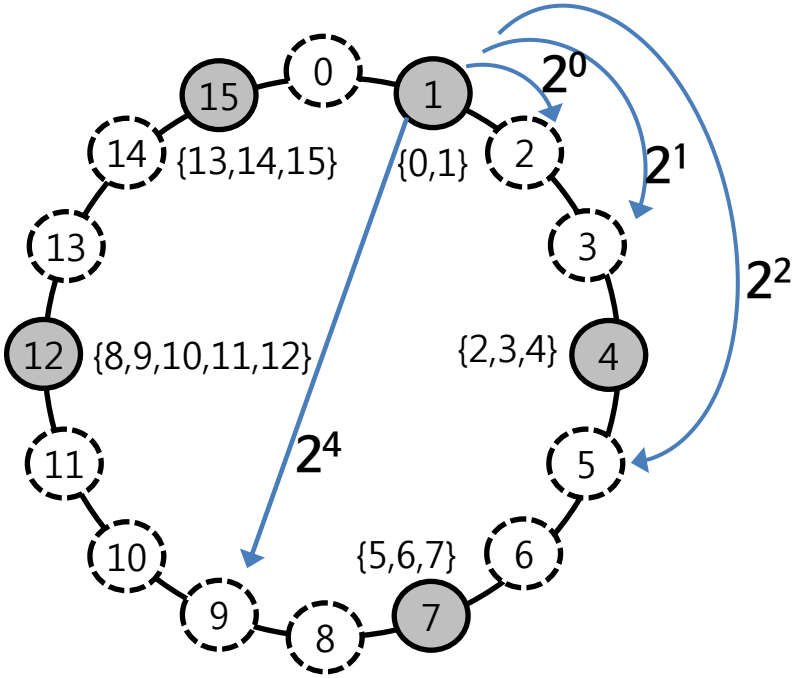
Chord DHT

Routing Algorithm

Finger Table

- 자신으로부터 2^k 번째 key의 successor
- ex) 노드1 의 Finger Table

k	2^k	successor(2^k)
0	1+1	4
1	1+2	4
2	1+4	7
3	1+8	12



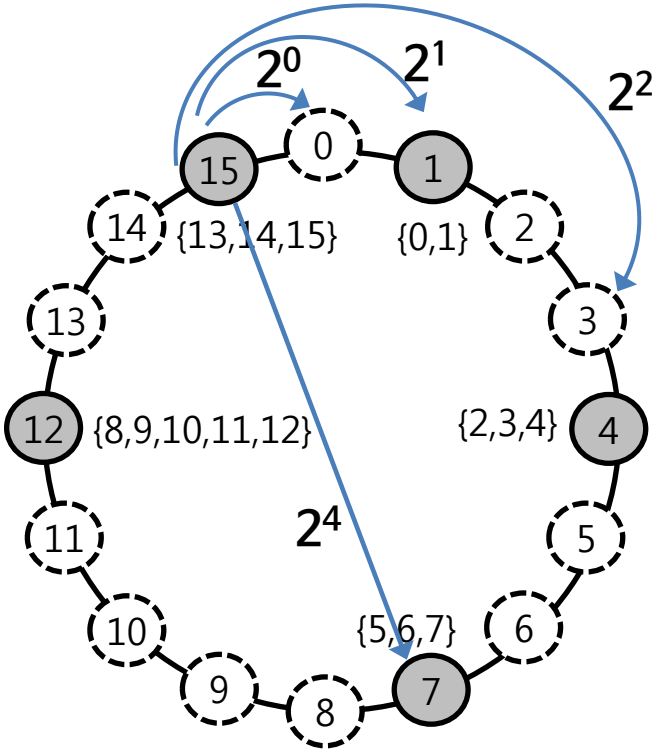
Chord DHT

Routing Algorithm

Finger Table

- 자신으로부터 2^k 번째 key의 successor
- ex) 노드15 의 Finger Table

k	2^k	successor(2^k)
0	(15+1)%16	1
1	(15+2)%16	1
2	(15+4)%16	4
3	(15+8)%16	7



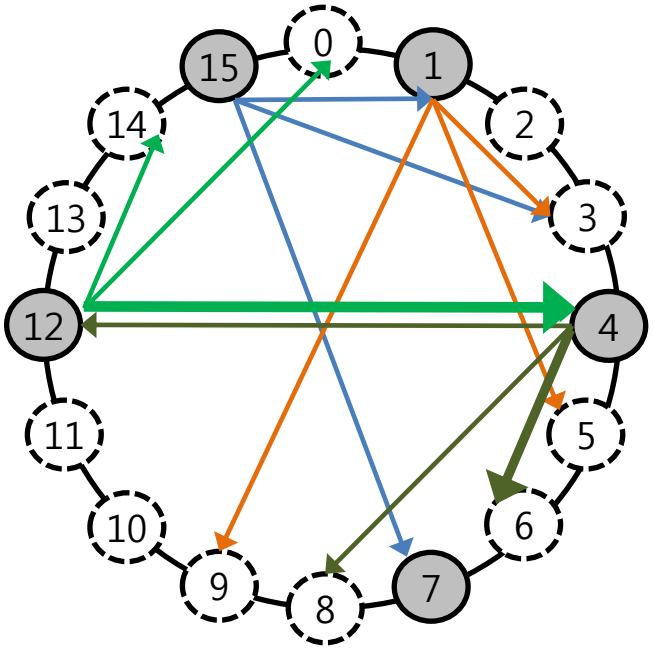
Chord DHT

Routing Algorithm

Decentralized Binary Search

특징

1. Decentralized Data Structure
2. 결함 내성
3. 확장성 (P2P)
4. 빠른 데이터 검색



EclipseMR의 아키텍처

이중 DHT 구조

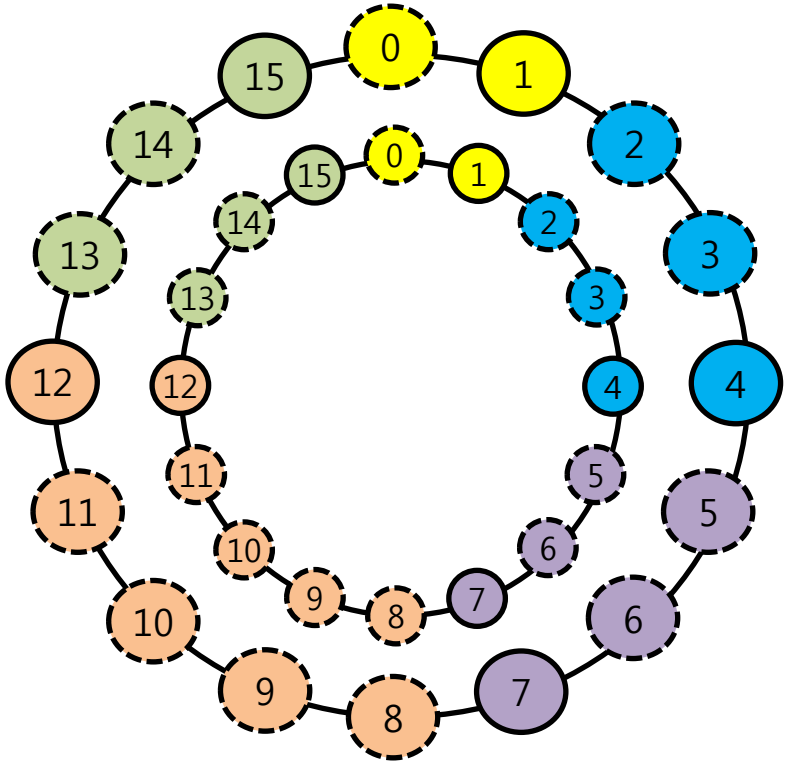
바깥 DHT: 인메모리 캐시

안쪽 DHT: 분산 파일 시스템

Key Design Principle

1) 로드 밸런싱

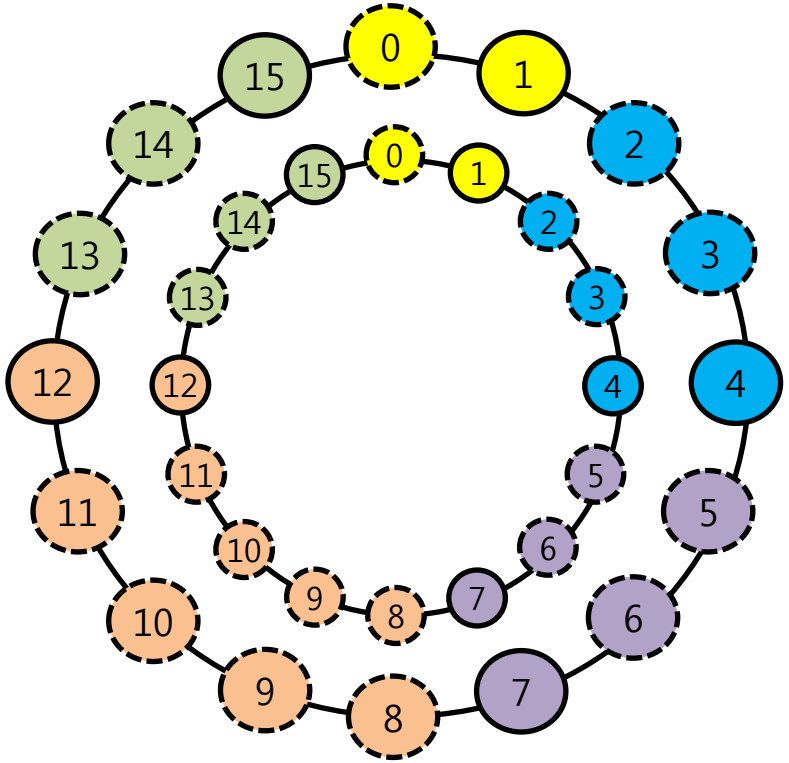
2) 인메모리 캐시 적중율



EclipseMR의 아키텍처

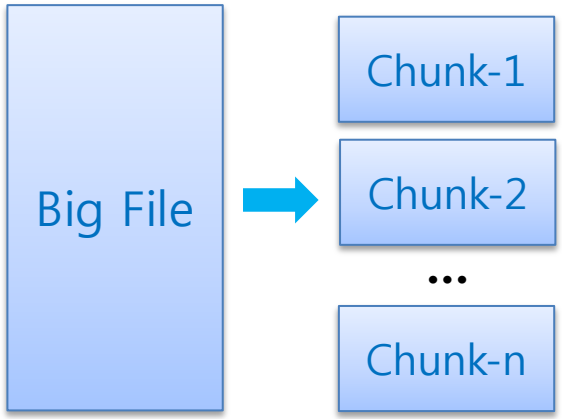
File Upload

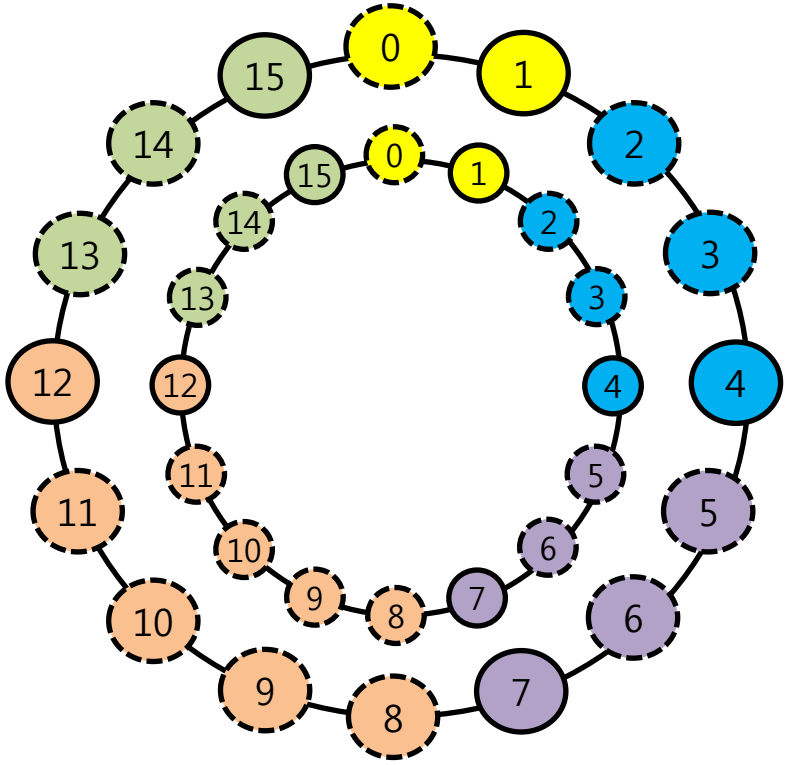
1) 128MB 청크로 분할
2) 분할된 청크의 Hash Key를 사용하여 저장될 서버 지정
3) Replication
   - replication factor = 3
   - 지정된 서버의 좌우 actual node에 저장



EclipseMR의 아키텍처

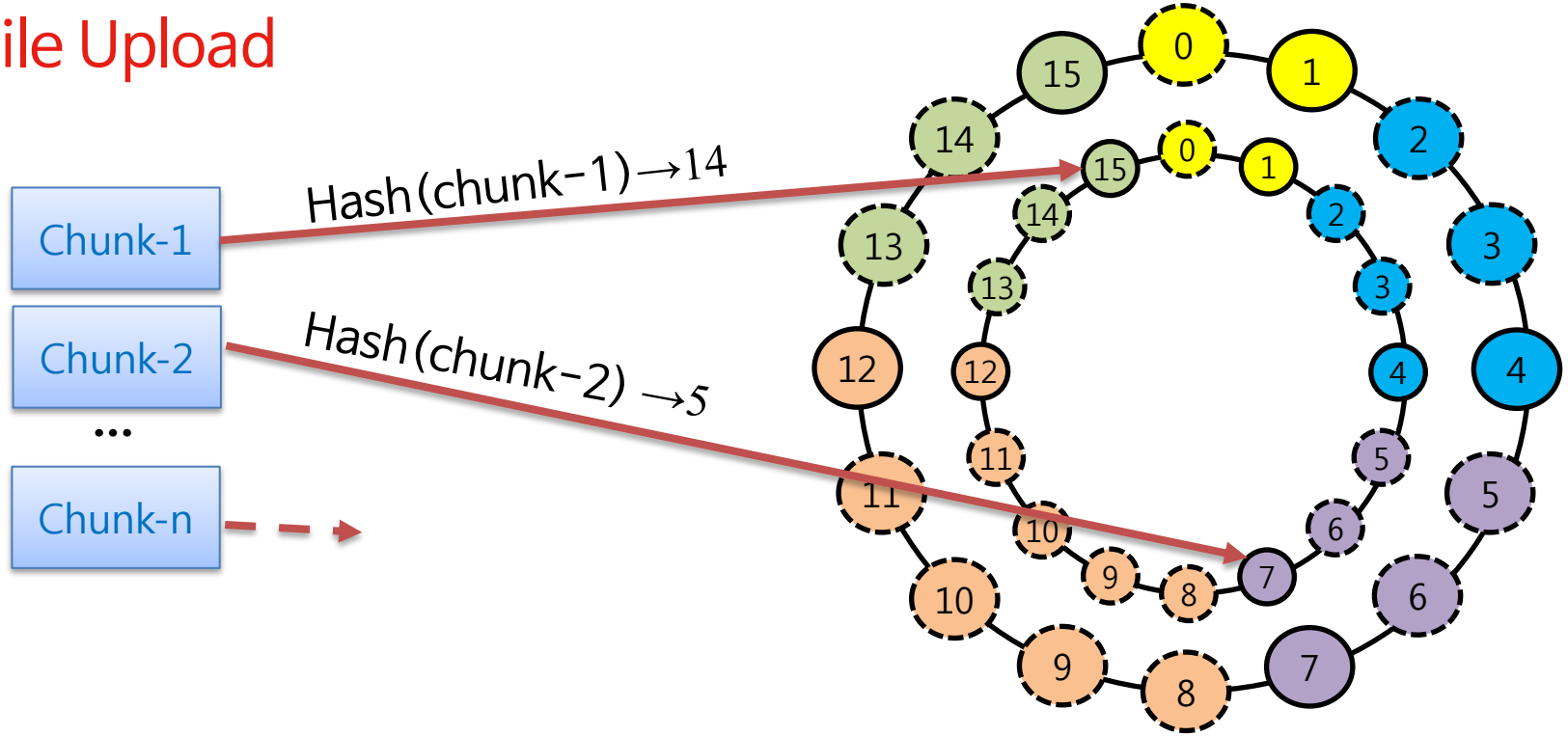
File Upload





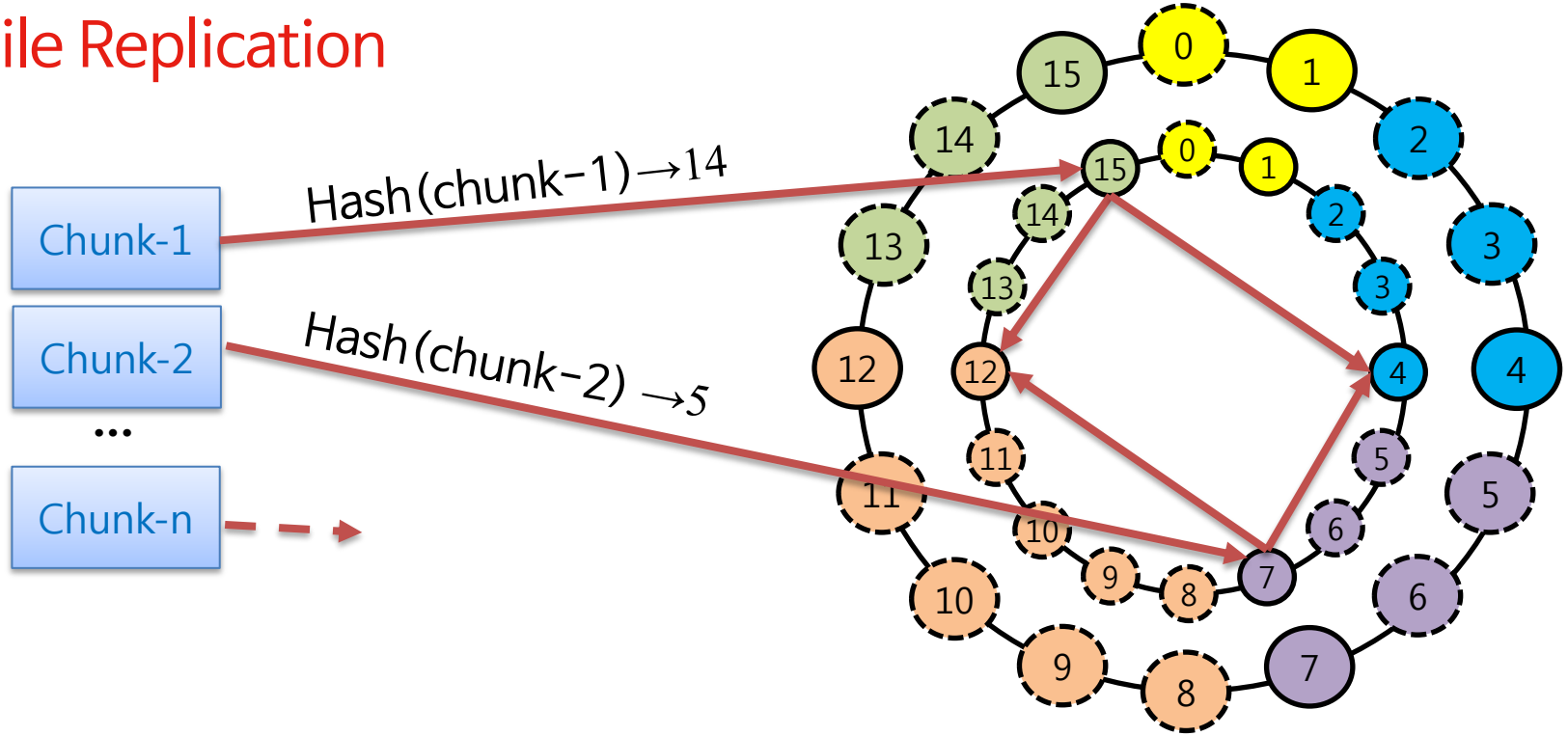
EclipseMR의 아키텍처

File Upload



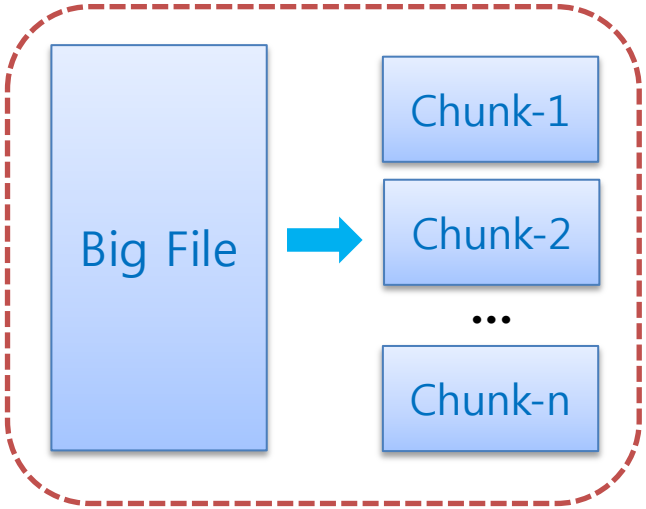
EclipseMR의 아키텍처

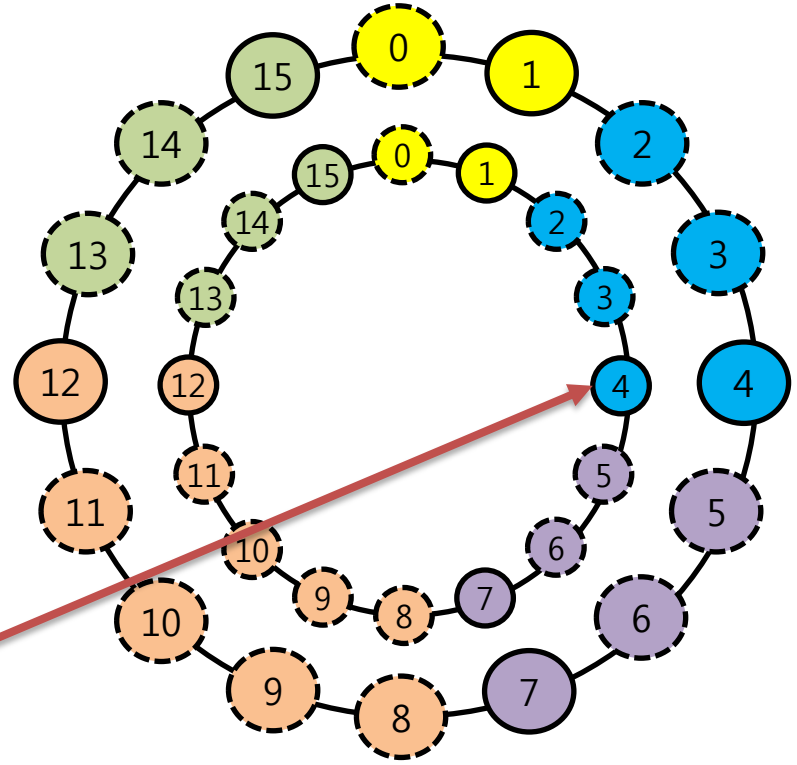
File Replication



EclipseMR의 아키텍처

File Metdata Decentralization



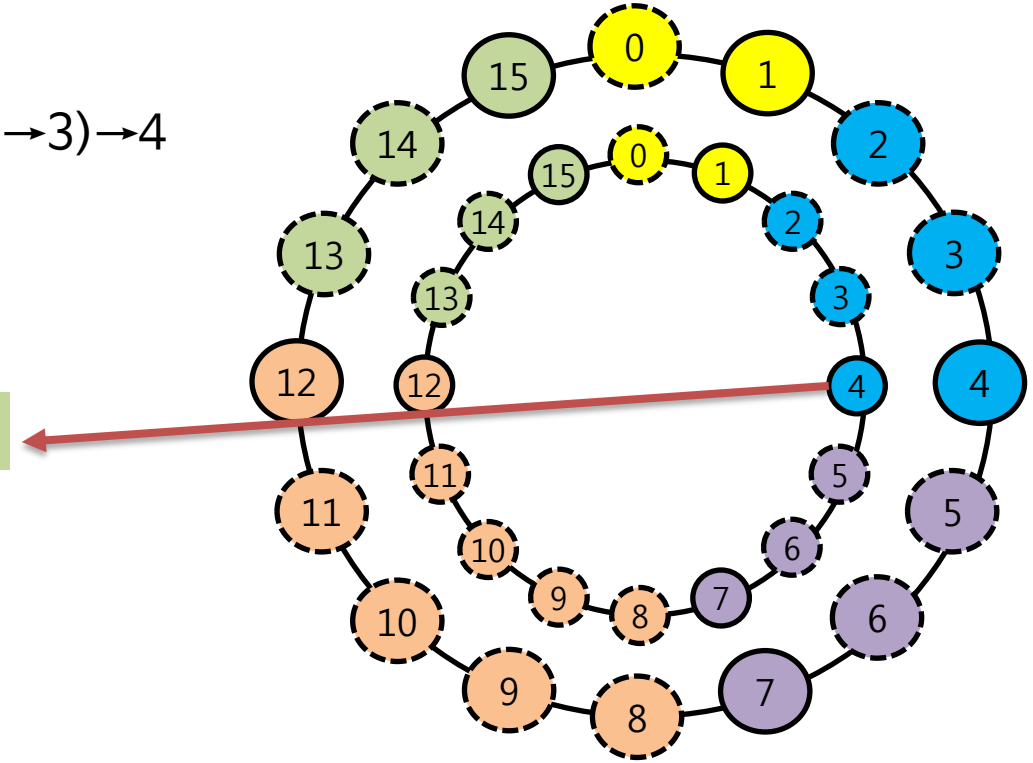


Partitionining/Replication 메타데이터
: Hash(Big File)→3

EclipseMR의 아키텍처

File Read

1) successor(Hash(Big File)→3)→4
2) 노드 4에서 파일 정보 획득
   - Chunk 갯수, ACL
     분할 정보,복제정보, etc
     BigFile 메타데이터
3) Hash(Chunk-1)→14
   Hash(Chunk-2)→5



EclipseMR의 아키텍처

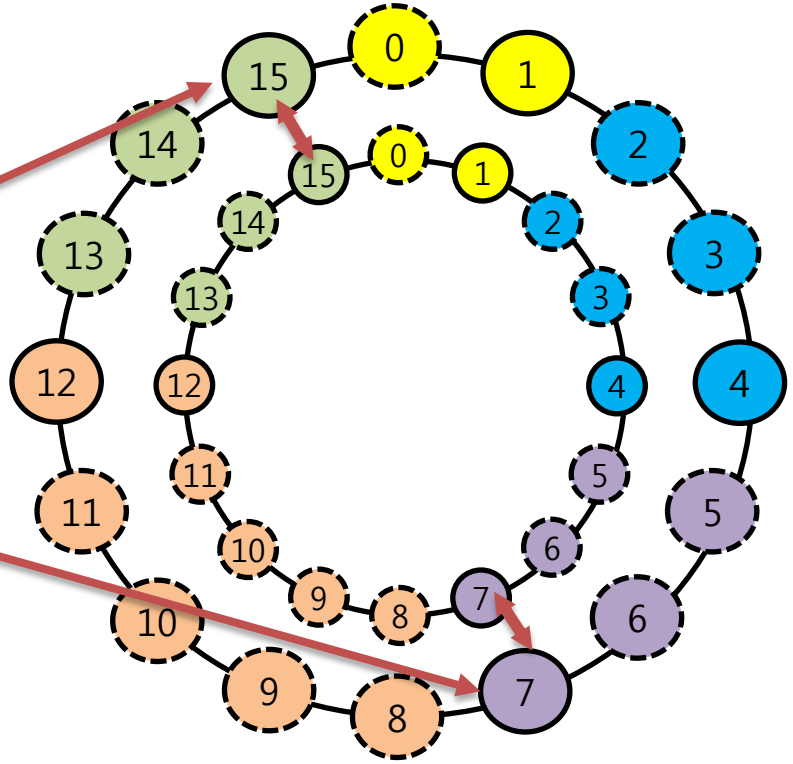
File Read

4) Search_InMemory(Hash(Chunk))

- 인메모리 캐시에서 히트 발생 기대

Chunk-1

Chunk-2



5) If Cache Miss → DHT 파일시스템에서 읽어 캐시에 저장

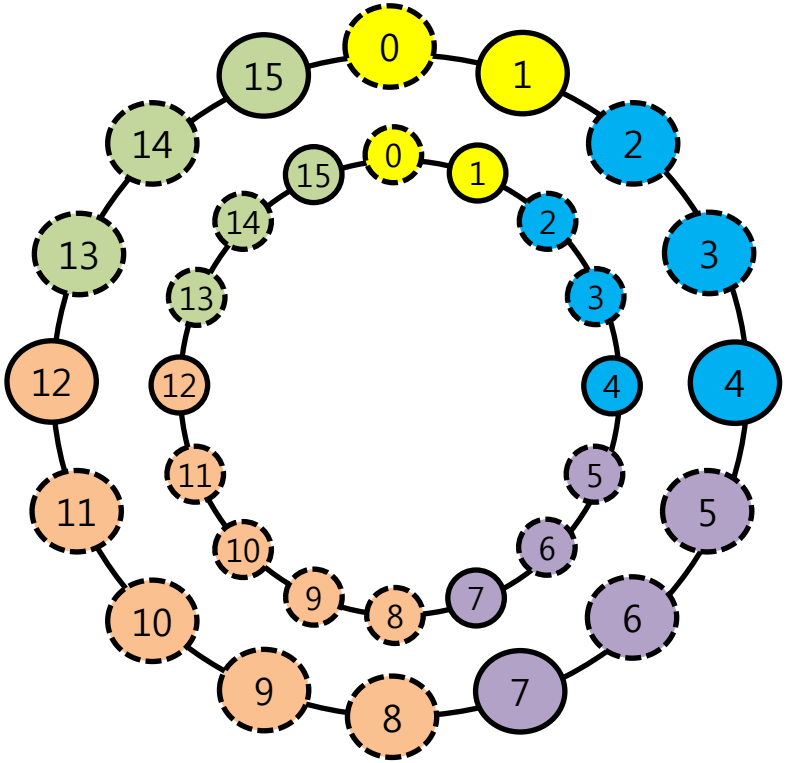
EclipseMR의 아키텍처

Load Balancing

Q: 만약 Hash 키 13,14,15 가 다른 키에 비해 무수히 많이 엑세스 된다면?

A1: Migration

A2: Replication



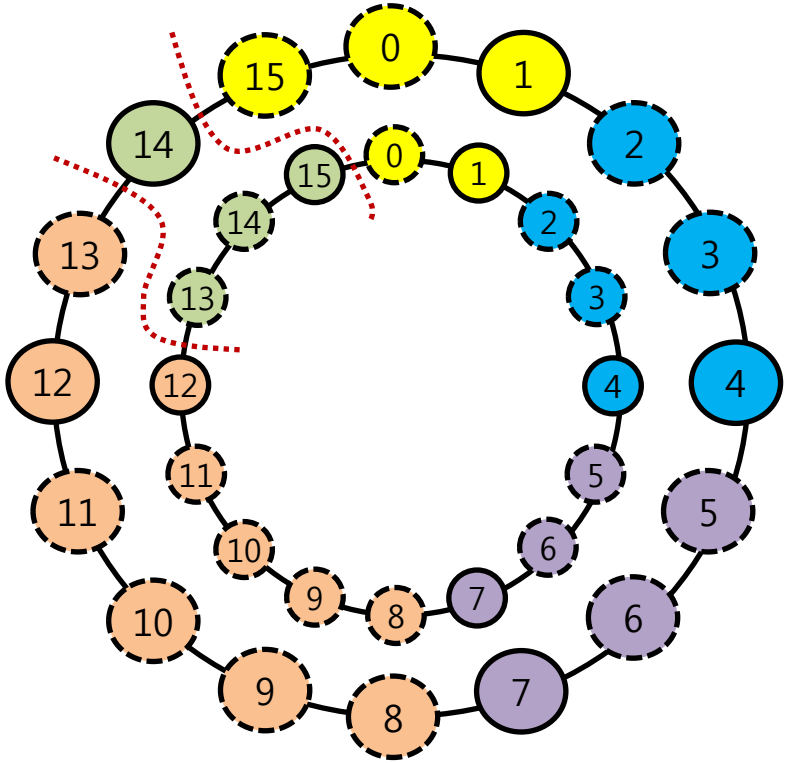
EclipseMR의 아키텍처

Load Balancing

Cache Migration/Replication:

- 인메모리 캐시의 데이터를
  이웃 노드로 복제 혹은 이동

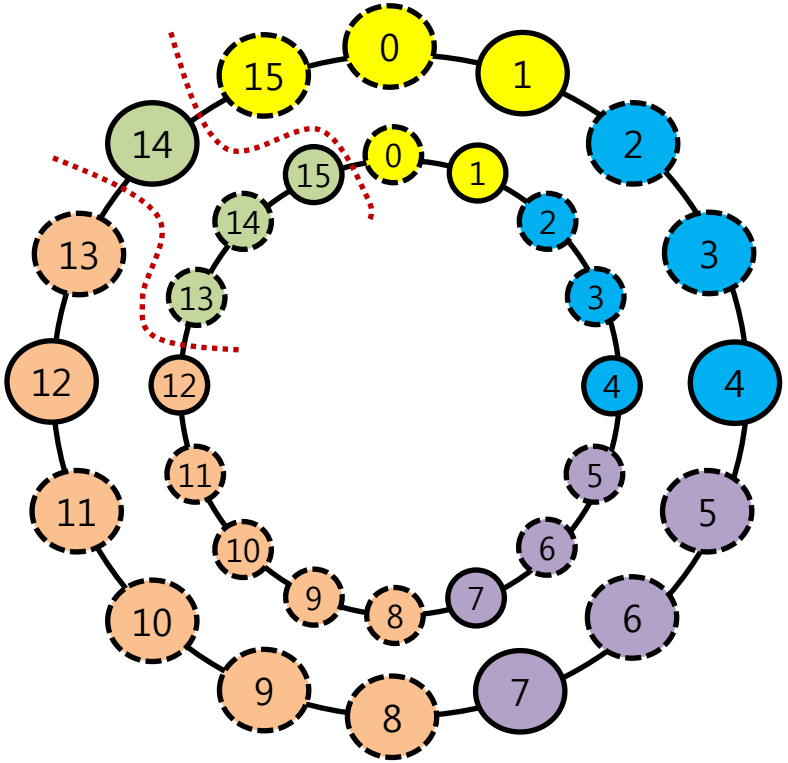
- 마이그레이션 알고리즘:
  - Locality-aware Fair 스케줄러
  - 인메모리 캐시의 Hash 경계선을
    움직여 로드밸런싱 유지



EclipseMR의 아키텍처

Remote Data Access

- 만일 Application이 key13을 읽기 위해 노드 12의 인메모리 캐쉬 검색.
- Cache Miss 발생 시 로컬디스크가 아닌 이웃한 노드 15에서 읽어야함
  → 성능 저하
- But! 15에 저장된 데이터들은 12에 복제 되어 있어 Local에서 읽음
- 경계가 극단적으로 변하지 않는 이상 로컬리티 보장

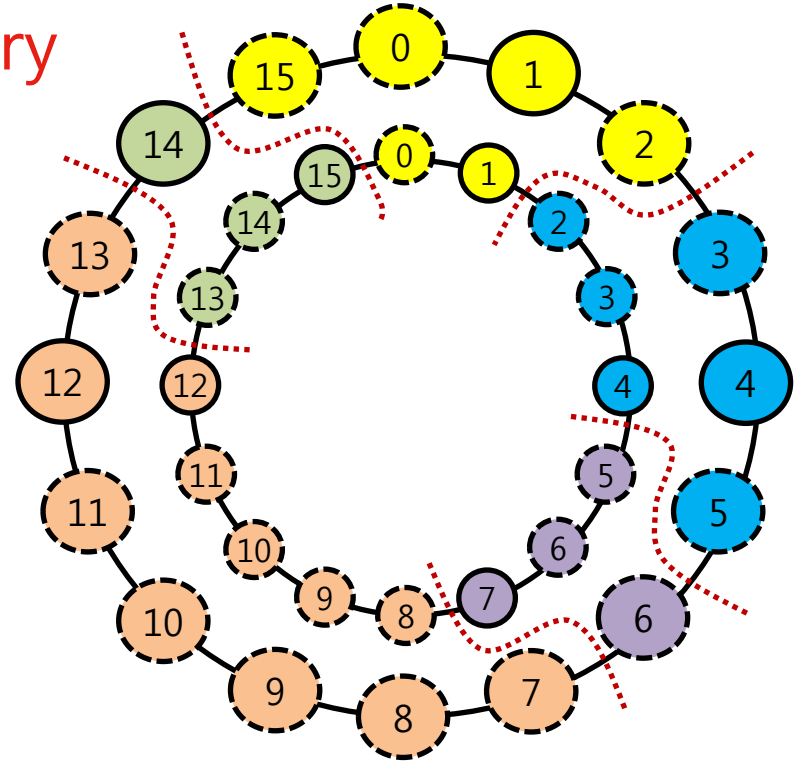


EclipseMR의 아키텍처

In-Memory Hash Key Boundary

- 로드밸런싱을 위해 Hash Key 경계선을 움직일 경우, Hash Key 경계선의 Decentralization이 어려움

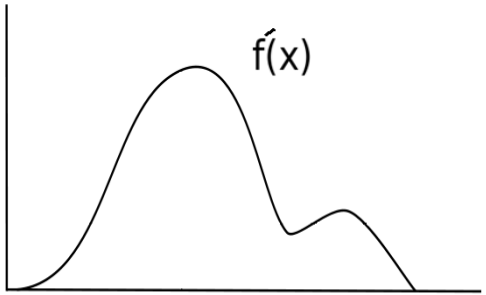
- Leader Election 을 통해 Hash Key 경계선과 Job Scheduling을 담당할 Frontend 노드를 P2P로 결정

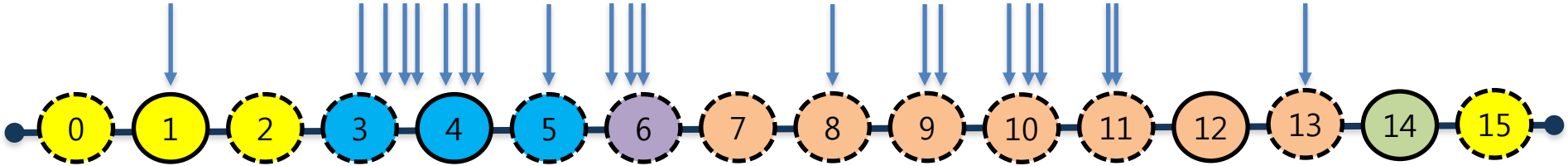


Locality-aware Fair Scheduler

In-Memory Hash Key Boundary Management

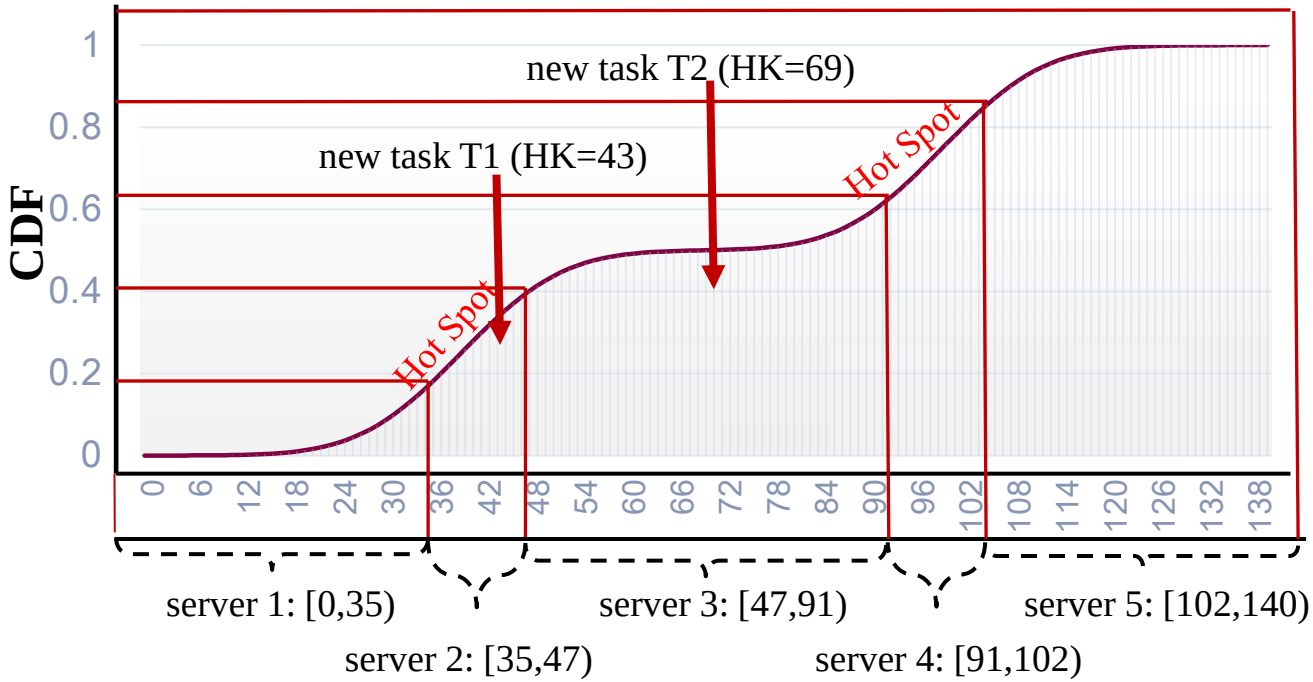
- 최근 데이터 액세스 분포를 사용하여 Hash Key 경계를 분할



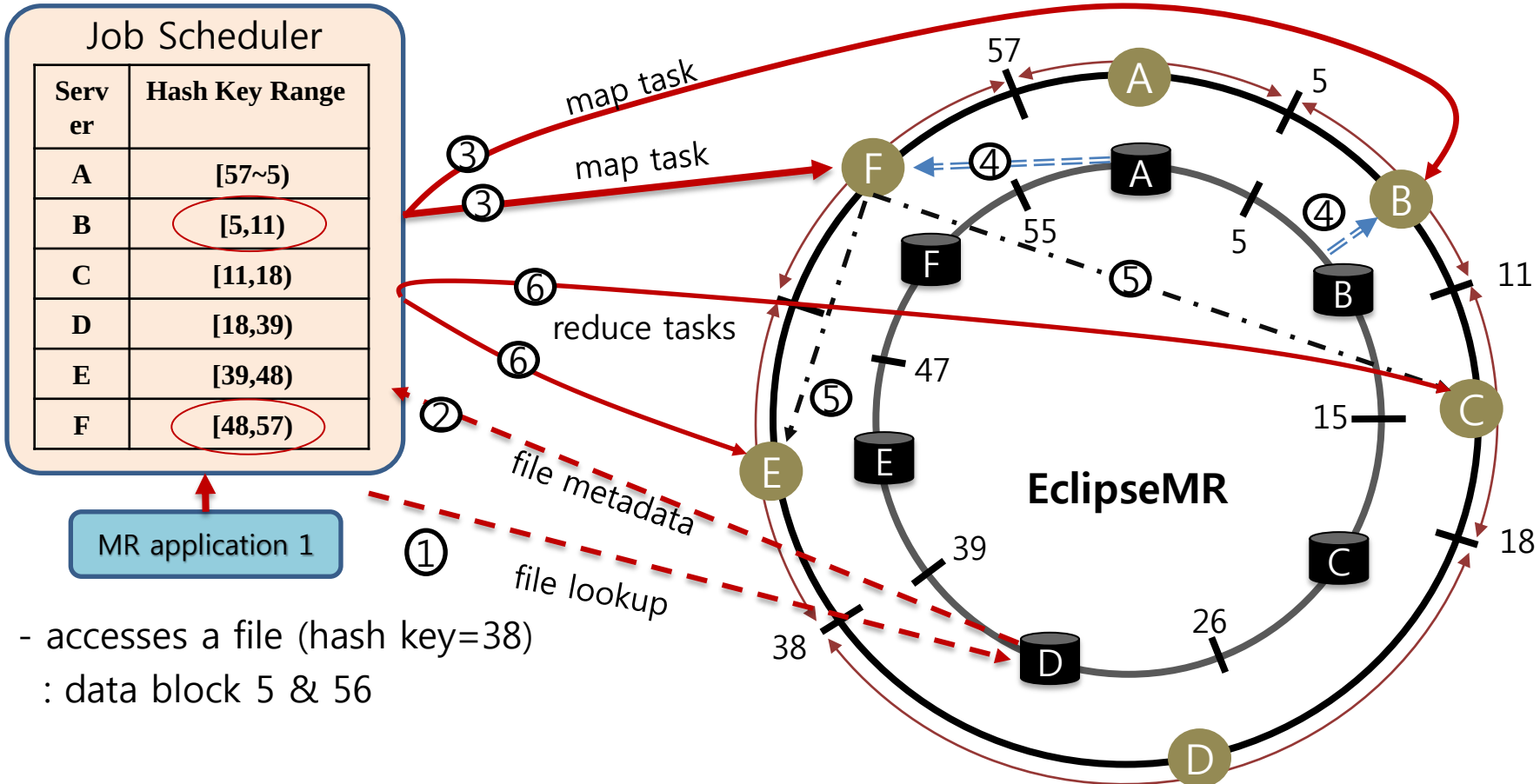


Locality-aware Fair Scheduler

In-Memory Hash Key Boundary Management

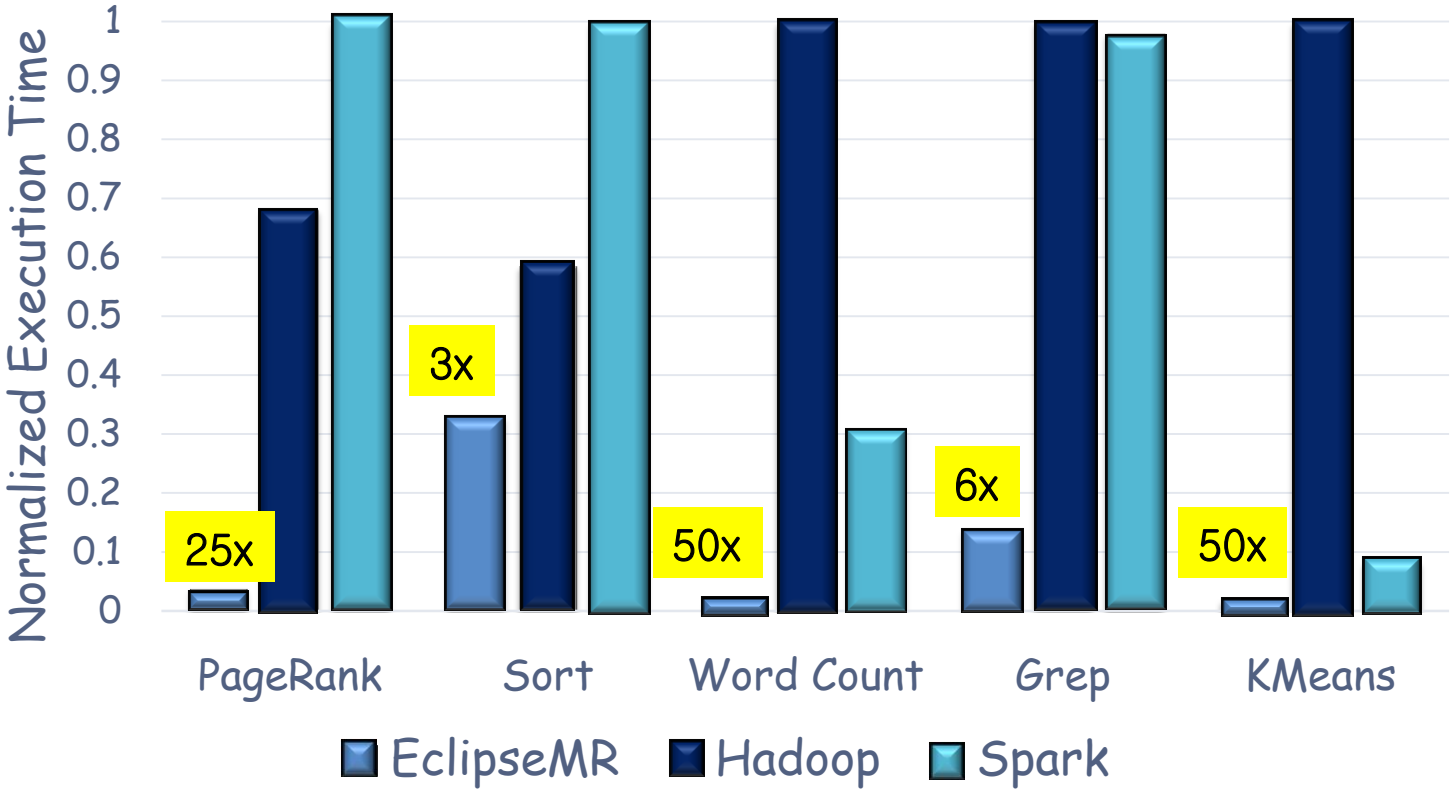


EclipseMR 에서의 MapReduce



- accesses a file (hash key=38)
  : data block 5 & 56

EclipseMR vs Hadoop & Spark

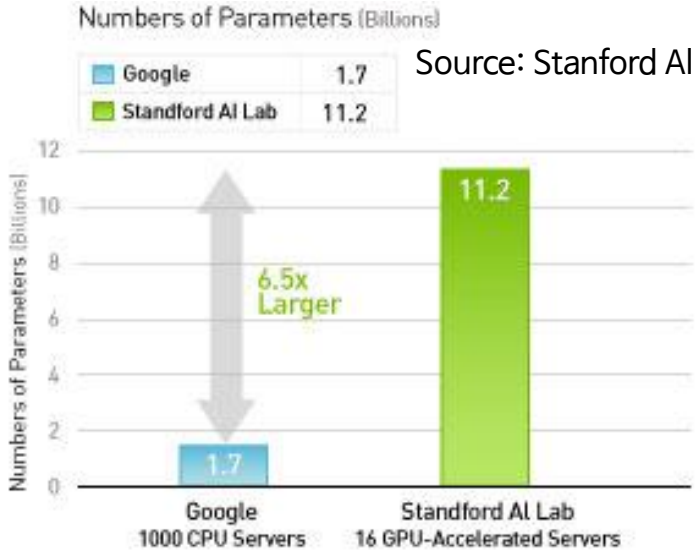


3.
맺음말

Ongoing Works

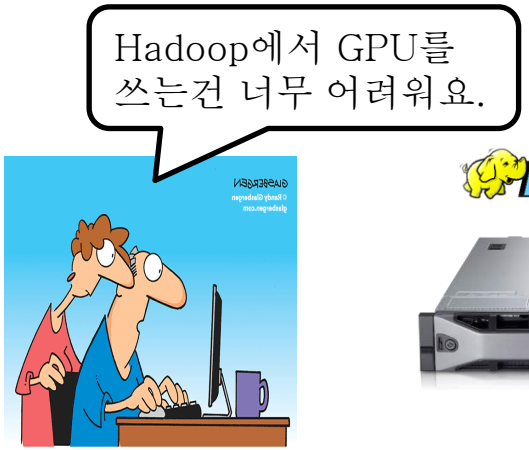
MachineLearning on EclipseMR

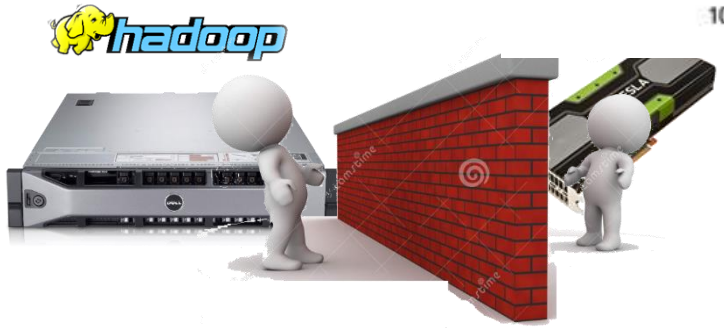
- GPU ML 패키지와 EclipseMR의 통합



Source: Stanford AI Lab

WORLD'S LARGEST ARTIFICIAL NEURAL NETWORKS WITH GPU



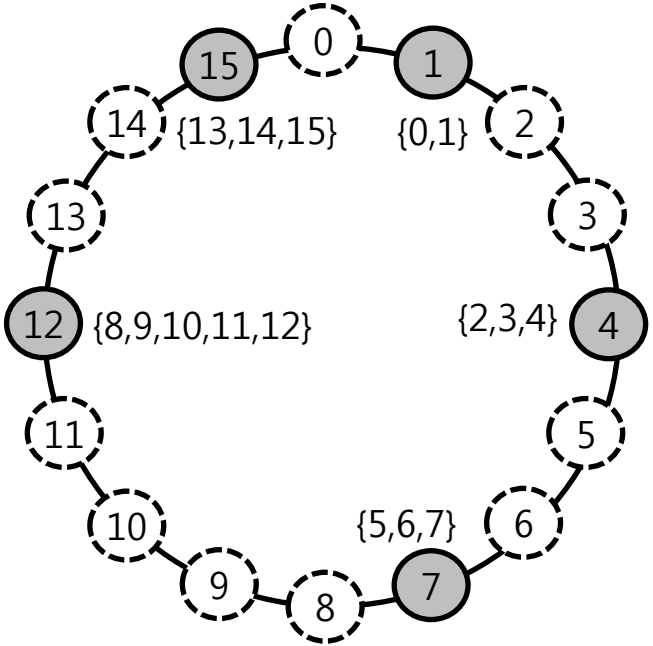


Ongoing Works

Cassandra/NoSQL

- Cassandra 와 EclipseMR의 통합

- NewSQL on EclipseMR
  DHT 인메모리 캐싱/RDMA 를 활용한
  고속 NewSQL 엔진 개발



Ongoing Works

MPI/RDMA

- 수퍼컴퓨팅 SGE/SLURM 환경에서 수행 가능

Supercomputer

빅데이터처리

맺음말

HPC & Big Data는 공통 과제를 함께 해결

- HPC의 Scale-up technologies를 BigData에 우선 적용
- UNIST 수퍼컴퓨팅센터/초고성능 빅데이터 사업단
  - EclipseMR 및 다양한 HPC + BigData 연구 중

Q&A

Thank You