IRIS Conecept Guide

Mobigen, November 20, 2015

목차

1 지원 정보

1.1 문서 버전

0.1

1.2 IRIS 버전

1.5.1

1.3 사용 문의 및 기술 정보

(주) 모비젠

본사

- 135-280 서울 강남구 대치동 967-3번지 KM빌딩 2층, (주) 모비젠
- T : 02 - 538 - 9360
- F : 02 - 538 - 9369

기술 연구소

- 135-280 서울 강남구 대치동 967-3번지 KM빌딩 5층, (주) 모비젠
- T : 02 - 538 - 9364
- F : 02 - 538 - 9368

모비젠 IRIS 기술 지원

- T : 02 - 538 - 9364
- M : iris@mobigen.com

2 IRIS Overview

본 항목에서는 IRIS 시스템의 기본 개념에 대하여 설명합니다.

2.1 Overview

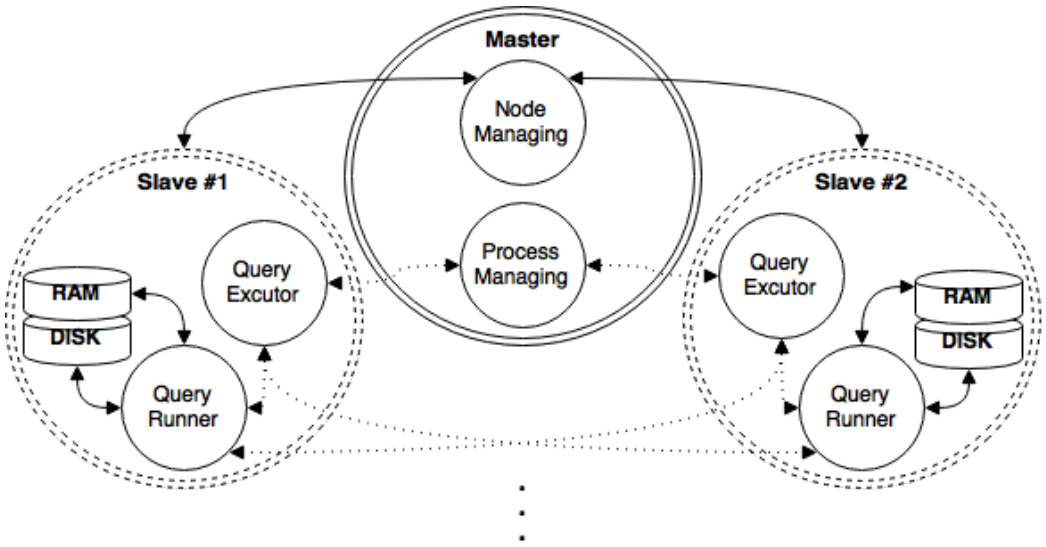

그림 1: IRIS Overview

IRIS는 SQL을 지원하는 분산 아키텍쳐 기반의 Big Data 데이터 베이스 솔루션 입니다. 단일 시스템으로 구성되어 있는 일반 DBMS와 다르게 IRIS는 쿼리에 대한 데이터 및 처리 프로세스의 분산을 관리하는 마스터 노드와 분산된 쿼리를 수행하는 슬레이브 노드로 구성됩니다. IRIS는 슬레이브 노드의 확장 및 증설이 용이하고, 데이터 저장 공간으로 메모리와 디스크를 모두 사용하는 하이브리드 (In-Memory, On-Disk) 방식의 데이터 베이스 입니다.

2.1.1 IRIS, 주요 지원 항목

- Linux 기반, IRIS의 모든 노드는 서버에서 널리 사용되고 있는 Linux를 최적화 커스터 마이징 하여 사용합니다.
- SQL 지원, IRIS는 자체적으로 SQL을 지원하여 SQL을 아는 모든 사용자 및 모든 개발자들이 간단한 교육만으로 쉽게 사용할 수 있는 환경을 제공합니다.
- ClientPackage, IRIS는 IRIS 전용 CLI와 데이터베이스 접근에 적용되는 JDBC를 비롯한 각종 API (JAVA, C, C#, Python)를 포함한 패키지를 제공합니다.

2.2 IRIS의 특징

2.2.1 실시간 (Real-Time) 통계 / 분석

IRIS의 슬레이브 노드에서는 최근 시점에 입력된 데이터들을 사용자가 정의한 일정 기간 동안 메모리상에서 관리합니다. 따라서 메모리상의 데이터들을 대상으로 고속의 조회/통계/분석 처리가 가능합니다.

2.2.2 대용량 데이터의 처리

IRIS는 분산 처리 구조를 채택하고 있으므로 PetaByte(1,000TB) 수준의 대용량 BIG Data의 관리가 가능합니다. 수십 TB 수준의 데이터를 관리하는 수십대의 슬레이브 노드를 하나의 시스템으로 구성했기 때문입니다. 따라서 성능, 가격 등의 문제로 기존에는 처리가 불가능했던 대용량의 Big Data들을 IRIS를 이용하여 분석이 가능합니다.

2.2.3 Scale-Out, 확장성

IRIS는 Scale-Out이 가능한 분산 구조로서 용량의 증설, 슬레이브 노드의 추가에 제한을 받지 않습니다. 모든 데이터의 다중화를 통하여 IRIS 확장 작업 중에도 지속적인 서비스가 가능합니다.

2.2.4 Fault-Tolerance, 무정지 서비스

IRIS로 저장되는 데이터는 최소 2개 이상 (기본 설정이 2개) 슬레이브 노드로 중복 저장됩니다. 이러한 데이터의 이중화를 통하여 일부 슬레이브 노드의 장애 상황에서도 서비스를 유지할 수 있으며, 데이터 손실 및 손상을 방지할 수 있습니다. 또한 슬레이브 노드의 부하를 균등하게 유지합니다. 마스터 노드는 Active / Standby 이중화를 통하여 무정지 서비스를 제공합니다.

2.2.5 분산 / 전역 테이블 간의 JOIN 지원

일반적인 Big Data 시스템은 노드간의 데이터 공유를 하지 않는 "Shared Nothing" 개념을 따릅니다. Shared Nothing 구조인 Big Data DB에서 Join 연산은 그것 자체로 매우 중요한 이슈입니다. IRIS에서는 분산 저장되어 있는 Big Data Table 간에 Join 연산을 지원하지는 않지만, 일반적인 관리 S/W들이 필요로하는 Join을 지원하기 위해서 분산된 Big Data Table, 즉 Local 테이블과 Global 테이블이라는 개념을 도입하여 Local / Global 테이블 간의 Join을 지원합니다.

- Local 테이블 : 슬레이브 노드에 분산되어 관리되는 테이블입니다. 각 테이블의 데이터는 모든 슬레이브 노드에 분산되어 저장되므로 Local 테이블 간의 Join은 지원하지 않습니다.

- Global 테이블 : 마스터 노드, 슬레이브 노드에 모두 동일한 데이터가 유지되는 테이블 입니다. 이 테이블은 모든 노드에 동일한 데이터를 가지고 있으므로 Join을 지원합니다.

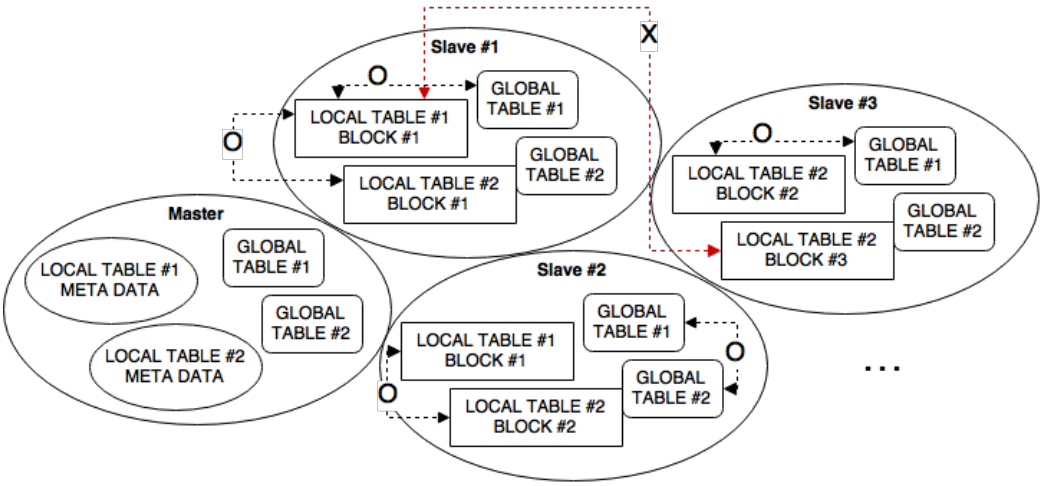

그림 2: Global & Local Table Join

위의 그림은 Global, Local 테이블간의 Join 가능성을 표현한 그림입니다. 먼저 Global 테이블간의 Join의 가능성입니다. 위의 그림을 참고하면 아래와 같은 결과가 나타납니다.

```
Global Table #1 & Global Table #2 : O
```

이므로 Global Table 간의 Join은 가능합니다. 다음은 Global Table 1과 Local Table 1 간의 Join 입니다.

```
Global Table #1 & Local Table #1 Block #1 : O
Global Table #1 & Local Table #1 Block #2 : O
```

다음은 Local Table 1과 Local Table 1간의 Join 입니다.

```
Local Table #1 Block #1 & Local Table #2 Block #1 : O
Local Table #1 Block #1 & Local Table #2 Block #2 : O
Local Table #1 Block #1 & Local Table #2 Block #3 : X
```

이므로 Join 불가입니다.

2.3 IRIS 구성

IRIS 시스템은 마스터 / 슬레이브 구조를 가지고 있으며 데이터 처리를 위한 다양한 프로세스들이 서로 연결되어 동작하고 있습니다. 다음 그림은 IRIS 시스템에 대한 프로세스 구성 요소를 표현한 그림입니다.

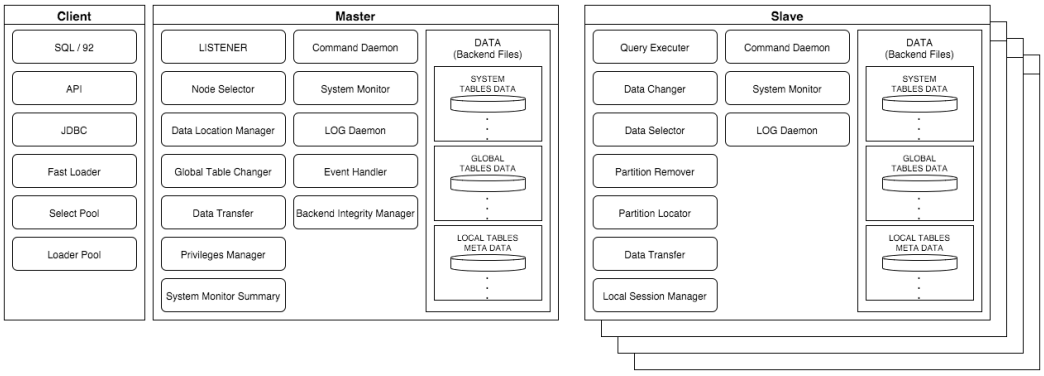

그림 3: IRIS S/W Architecture

2.3.1 주요 프로세스 [노드 모니터링]

2.3.1.1 Event Handler [EHD] 각각의 노드에서 발생하는 이벤트와 각 노드의 System Monitor 로부터 발생하는 상태 정보를 바탕으로 노드 상태를 확인 / 관리하고 장애 발생시 노드의 상태를 변경합니다. 또한 사용자가 확인할 수 있는 메시지를 생성합니다.

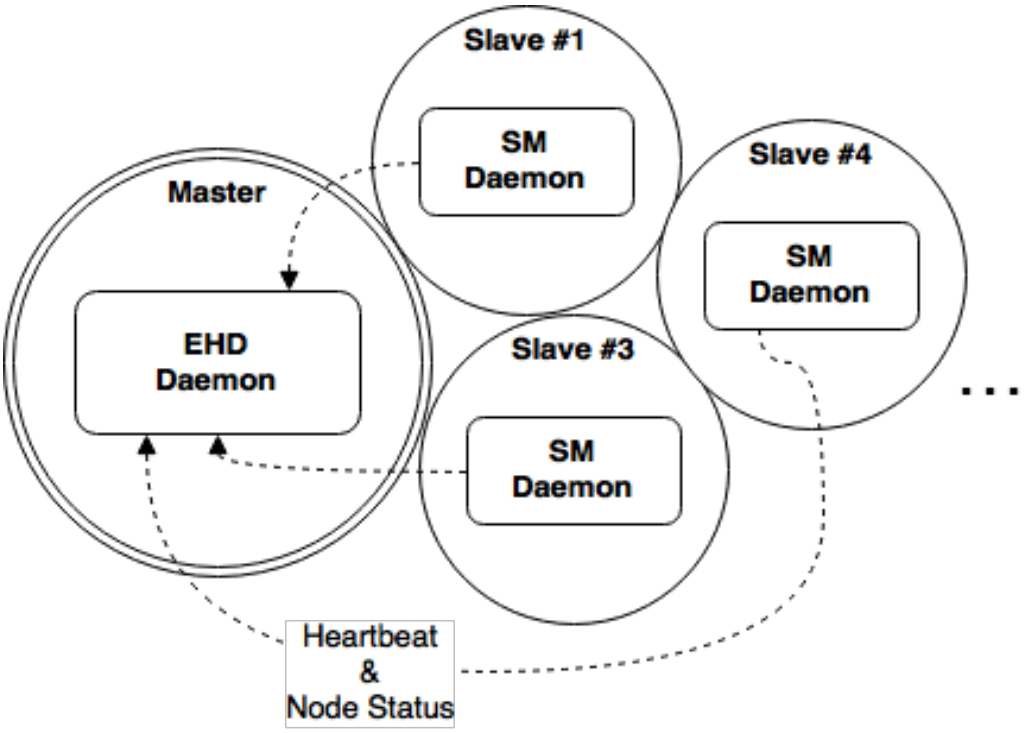

그림 4: IRIS Node Monitoring

2.3.1.2 System Monitor [SM] 실행된 각 노드의 시스템 정보, 테이블 정보, 상태 정보를 주기적으로 Event Handler로 전송합니다.

2.3.2 주요 프로세스 [쿼리 실행]

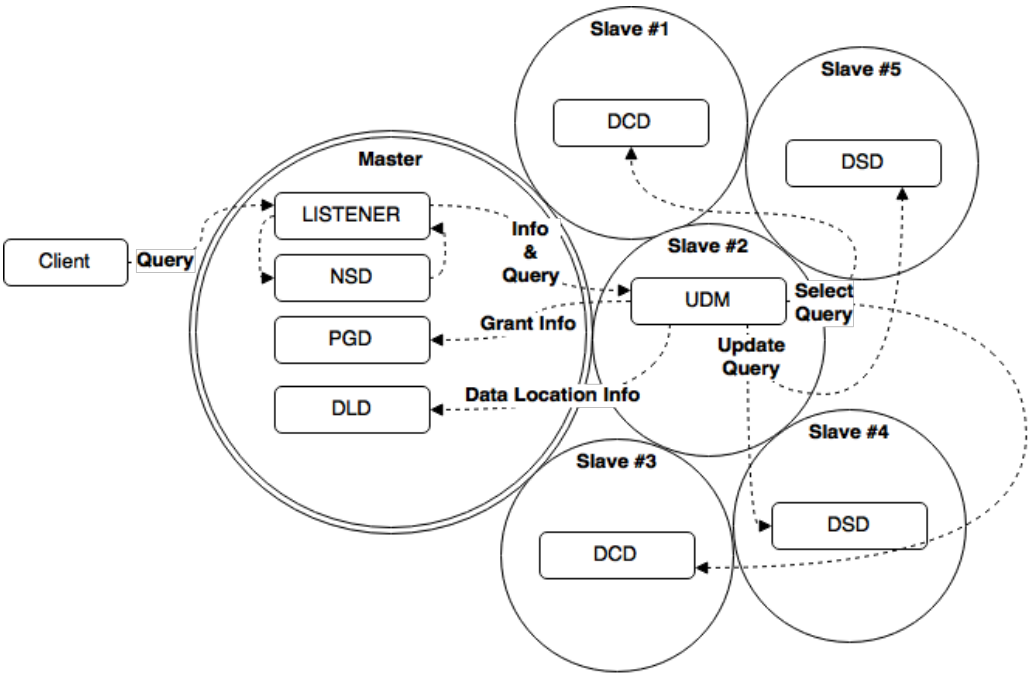

그림 5: IRIS Run Query

2.3.2.1 LISTENER [LISTENER] 클라이언트의 접속 요청을 받아들이며, Node Selector와 내부 통신하여 클라이언트의 쿼리 요청을 처리하기 위한 Query Executer를 할당합니다. 그 후 클라이언트와 Query Executer 간의 데이터 전달 역할을 수행합니다.

2.3.2.2 Node Selector [NSD] 클라이언트 작업 요청을 처리하기 위한 Query Executer를 할당합니다.

2.3.2.3 Data Location Manager [DLD] KEY / PARTITION 에 의해 분배되어 있는 데이터의 위치 정보, 데이터 상태 정보를 생성/조회/수정/관리하는 역할을 수행합니다.

2.3.2.4 Privileges Manager [PGD] 데이터의 접근 권한을 관리합니다.

2.3.2.5 Query Executer [UDM] 사용자 요청에 대한 작업을 수행하는 데몬으로, 사용자의 Query를 분석하여 작업 과정을 생성하고 작업 종류에 따라 Data Changer 혹은 Data Selector 에 데이터 처리 명령을 내리며, 그 결과들을 수집하여 최종 결과를 생성하는 여할을 담당합니다.

2.3.2.6 Data Changer [DCD] 실제 데이터가 저장된 Backend의 정보를 변경하기 위해 사용되는 데몬입니다.

2.3.2.7 Data Selector [DSD] Backend의 정보를 수집하기 위해 사용되는 데몬입니다.

2.3.3 주요 프로세스 [데이터 관리]

2.3.3.1 Partition Remover [PR] 데이터 삭제를 담당하는 프로세스입니다. Disk 보관주기가 지난 파일을 삭제하며, Lock, TMP 파일 등을 삭제하는 데몬입니다. 또한 깨진 파일을 복구합니다.

2.3.3.2 Partition Locator [PL] 데이터 위치를 관리하는 프로세스입니다. Ram, SSD 보관 주기에 따라 데이터 위치를 이동 시킵니다.

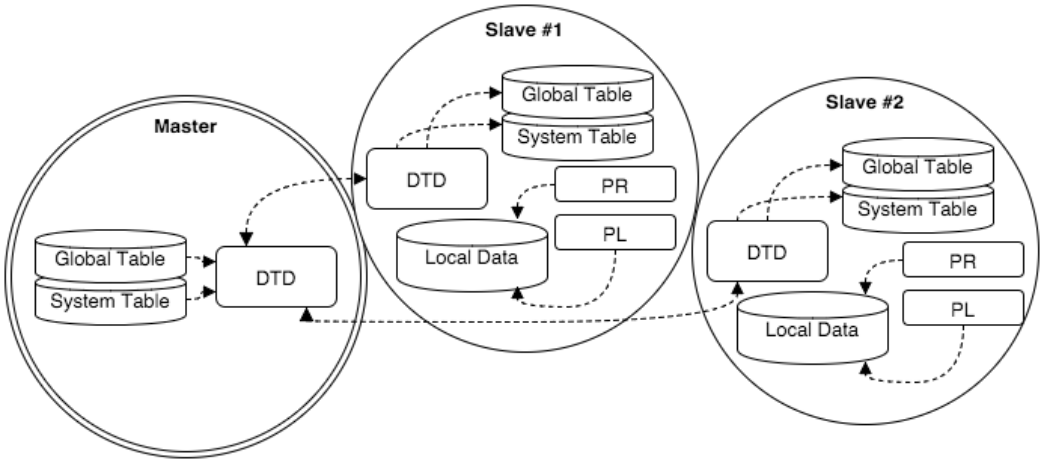

그림 6: Data Managing

2.3.3.3 Data Transfer DTD Global / System 테이블의 동기화를 위한 데몬입니다.

2.3.4 IRIS 디렉토리 구성

본 챕터에서는 IRIS 디렉토리 구성에 대해서 설명합니다. 아래 그림은 IRIS의 주요 디렉토리 구조입니다.

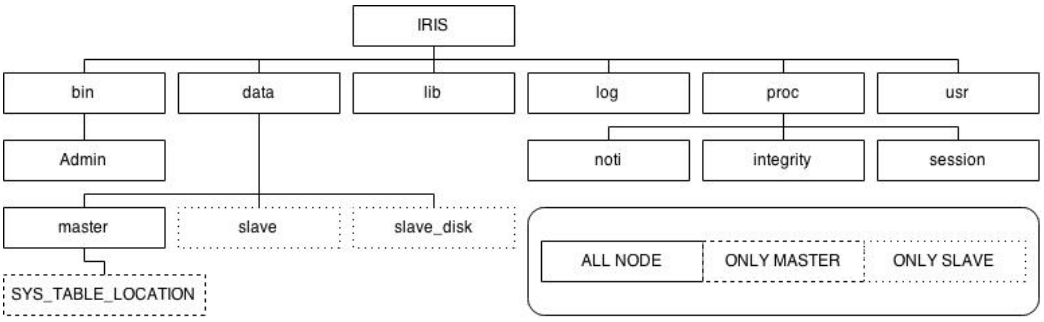

그림 7: IRIS Directory Architecture

가장 상위 폴더는 IRIS며, 그 아래에 bin, data, lib, log, proc, usr 폴더가

존재합니다. 각각의 폴더에 대한 설명은 다음과 같습니다.

2.3.4.1 bin 데몬 실행에 필요한 실행 파일들이 있는 디렉토리입니다.

2.3.4.1.1 Admin bin 디렉토리 밑에 있는 디렉토리로 IRIS를 관리하기 위한 실행파일들이 존재합니다. 대표적인 것으로 IRIS-Startup, IRIS-Shutdown, NodeEnable, NodeAdd 등이 존재합니다.

2.3.4.2 data IRIS에서 사용하는 데이터가 저장되는 디렉토리입니다. 하위 디렉토리로는 master, slave, slave_disk 등이 있으며 master 디렉토리는 마스터, 슬레이브 노드에 모두 존재하며 slave, slave_disk 디렉토리는 슬레이브 노드에만 존재합니다.

2.3.4.2.1 master master 디렉토리에는 System 테이블과 Global 테이블이 존재합니다. 하위 디렉토리에는 SYS_TABLE_LOCATION이 있습니다. SYS_TABLE_LOCATION 디렉토리는 마스터 노드에만 있으며 Local 테이블의 메타 데이터가 저장됩니다.

2.3.4.2.2 slave slave 디렉토리는 ramfs, 램 파일 시스템으로 램에 데이터를 저장하기 위한 공간입니다. IRIS에 처음 들어온 데이터는 이 곳에 저장됩니다. 이 디렉토리는 슬레이브 노드에만 존재합니다.

2.3.4.2.3 slave_disk slave_disk는 IRIS 슬레이브 노드에만 존재하는 디렉토리로 실제 데이터를 저장하는 디스크의 정보를 담고 있는 디렉토리입니다. 실제 사용하는 폴더의 링크가 저장되며 일반적으로 part00, part01, part02 … 형태로 링크가 생성되어 있습니다.

2.3.4.3 lib lib 디렉토리는 IRIS 코어 라이브러리를 포함하는 디렉토리입니다.

2.3.4.4 log log 디렉토리는 IRIS에서 발생하는 로그를 저장하는 디렉토리입니다.

2.3.4.5 proc proc 디렉토리에서는 IRIS에서 발생하는 이벤트를 저장합니다. 해당 디렉토리 아래에는 noti, session, integrity 정보를 저장하는 디렉토리가 존재합니다.

2.3.4.5.1 noti 특정 노드에서 발생한 이벤트를 저장합니다. 저장되는 파일명은 다음과 같습니다.

```
[이벤트 발생 시간]_[이벤트 발생 노드 IP].[이벤트 발생 등급]
```

마지막 확장자는 이벤트 발생 등급으로 아래와 같이 등급을 구분합니다.

- INFO : 단순 정보
- WARN : 주의 정보
- BUSY : 노드 상태가 BUSY로 변함을 알림
- FATAL : 위험 정보

2.3.4.5.2 session session 관련된 정보가 저장되는 디렉토리입니다.

2.3.4.5.3 integrity 데이터에 손상이 갔을 경우 손상된 데이터를 정상 데이터로 저장하기 위해 사용되는 정보를 저장하는 디렉토리입니다. 데이터 복제 수가 1일 경우에는 동작하지 않으며 데이터 복제 수가 2 이상일 경우 목록을 생성 저장하여 정상적인 데이터를 유지합니다.

2.3.4.6 usr IRIS Core 라이브러리를 동작시키는데 필요한 라이브러리를 저장한 디렉토리입니다.

2.4 IRIS 의 테이블 종류

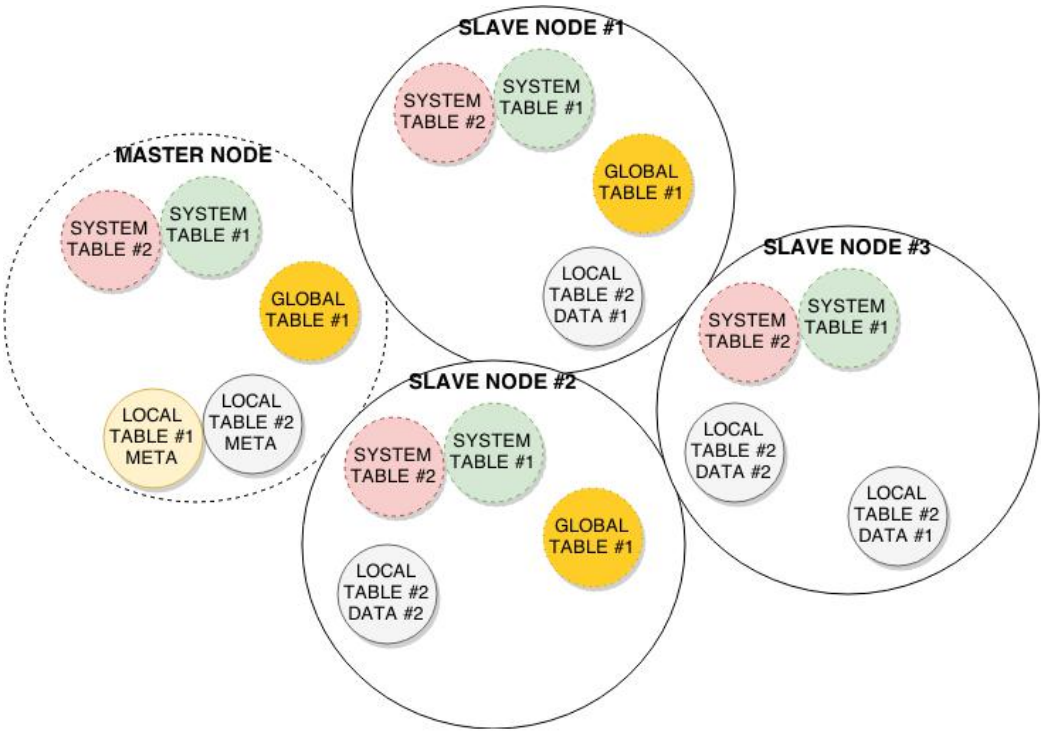

그림 8: 테이블 종류

IRIS의 테이블은 세 가지 종류로 나눌 수 있습니다. SYSTEM / GLOBAL / LOCAL 테이블이 그 종류입니다.

2.4.1 System 테이블

IRIS 시스템에서 사용되는 테이블로 IRIS를 구성하고 있는 정보를 담고 있는 테이블 입니다. 대표적인 예로는 노드 정보를 담고 있는 SYS_NODE_INFO, 사용자 정보를 담고 있는 SYS_USER_INFO 등이 있습니다. 모든 노드에 동일한 데이터가 항상 유지되는 테이블입니다.

2.4.2 Global 테이블

각각의 슬레이브 노드에서 동일한 데이터 접근이 가능한 테이블 입니다. 이 테이블은 주로 데이터 규모가 작고, 변경이 적게 일어나는 데이터에 한 하여 사용하기를 제안하고 있습니다. 데이터 조회는 슬레이브 노드에서 동작하며 데이터 수정은 마스터에서 수행된 후에 각각의 슬레이브 노드와 동기화 됩니다.

2.4.3 Local 테이블

대용량 데이터를 구성하는 경우에 사용되는 테이블입니다. 단일 파일로 기록되지 않고 다수의 슬레이브 노드에 KEY/PARTITION으로 구분된 여러 조각으로 나뉘어 저장됩니다. 마스터 노드에서는 해당 데이터의 위치 정보만 가지고 있으며 실제 데이터는 슬레이브 노드에 분산, 관리됩니다.

2.5 KEY / PARTITION

Key, Partition 은 IRIS 내부적으로 데이터를 분산 저장하기 위한 개념입니다. 즉, 분산되어 저장되는 Local 테이블에 적용되는 개념입니다.

KEY, 데이터베이스에서의 Key 개념과 다름

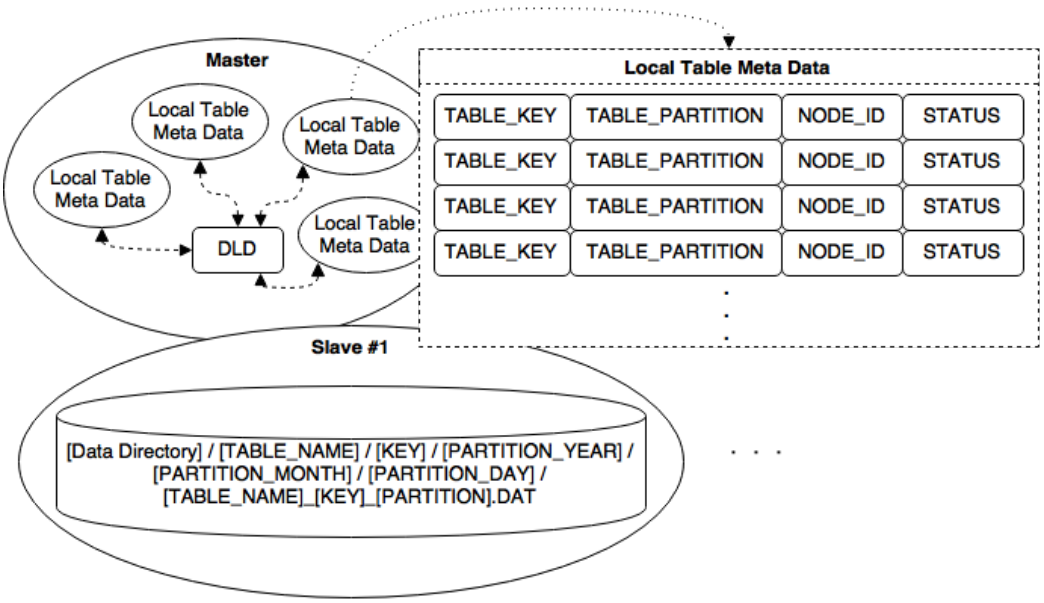

그림 9: Table Key Partition

Key, Partition 은 테이블 생성시 정의한 컬럼의 값을 따릅니다.

```
CREATE TABLE TEST_TABLE (
    A TEXT,
    B TEXT,
    C TEXT
)
datascope LOCAL
ramexpire 60
diskexpire 1440
partitionkey A
partitiondate B
partitionrange 10
;
```

위의 구조를 가지는 테이블 있다고 가정합니다. 먼저 “CREATE TABLE …. C TEXT)” 까지는 일반적인 CREATE TABLE 구문과 동일합니다. 하지만

하위에 옵션 값이 다른 것을 확인할 수 있습니다. KEY, PARTITION에 대해서 이해하고자 할 때 중요한 부분은 바로 partition으로 시작하는 옵션 값들입니다.

2.5.1 PARTITIONKEY

```
partitionkey A
```

테이블 생성시에 옵션으로 준 partitionkey A 라는 옵션이 바로 Key 컬럼을 설정하는 부분입니다. 특정 데이터의 Key 값은 바로 A 컬럼에 들어있는 값이 됩니다. 예를 들어

```
CAR,20150101000000,KIA
```

라는 데이터가 들어왔을 때 이 값은 Key가 CAR 인 곳에 저장됩니다.

2.5.2 PARTITIONDATE

```
partitiondate B
partitionrange 10
```

테이블 생성시에 옵션으로 준 partitiondate B 라는 옵션이 바로 Partition 컬럼을 설정하는 부분입니다. 특정 데이터의 Partition 값은 바로 B 컬럼에 들어 있는 값이 됩니다. 예를 들어

```
CAR,20150101000000,KIA
CAR,20150101000900,HYUNDAI
CAR,20150101001000,BENZ
```

라는 데이터가 차례대로 들어왔습니다. 먼저 첫 번째 데이터

```
CAR,20150101000000,KIA
```

값의 PARTITION 값은 20150101000000 입니다. 다음 데이터를 보겠습니다.

```
CAR,20150101000900,HYUNDAI
```

입니다. PARTITION 값을 확인하면 20150101000900 임을 확인할 수 있습니다. 하지만 partitionrange 10이라는 값을 확인할 수 있습니다. partitionrange 옵션은 몇분 단위로 PARTITION 값을 나눌것인가에 대한 옵션입니다. 즉, 20150101000900 는 partitionrange 10 옵션에 의해 최종적으로 PARTITION 값 20150101000900 이 됩니다.

```
partitionrange 10
20150101000000 - 20150101000959 : 20150101000000
20150101001000 - 20150101001959 : 20150101001000
20150101002000 - 20150101002959 : 20150101002000
.
.
.
```

위의 예제는 partitionrange 옵션이 10분일 경우에 어떻게 데이터가 저장되는지 볼 수 있는 리스트입니다.

2.5.3 KEY AND PARTITION

IRIS Local 테이블에 저장되는 모든 데이터는 Key, Partition 값을 참조하여 저장됩니다. 예를 들어

```
KIA,20150101000000,K5
HYUNDAI,20150101000000,SONATA
KIA,20150101000901,K3
HYUNDAI,20150101001002,AVANTE
```

라는 값이 있을 때

```
KIA,20150101000000,K5
KIA,20150101000901,K3
    => KEY : KIA, PARTITION : 20150101000000

HYUNDAI,20150101000000,SONATA
    => KEY : HYUNDAI, PARTITION : 20150101000000

HYUNDAI,20150101001002,AVANTE
    => KEY : HYUNDAI, PARTITION : 20150101001000
```

라는 결론이 만들어 집니다. 즉, 위의 두 개의 값은 같은 곳에 저장되고 아래의 값 두 개는 각각 다른 곳에 저장되어 짐을 알 수 있습니다.

다음은 KEY, PARTITION의 특징입니다.

- KEY 컬럼과 PARTITION 컬럼은 변경할 수 없습니다.
- Local 테이블에만 KEY/PARTITION 컬럼을 설정하며 Global 테이블에는 설정하지 않습니다. (설정은 가능하나 동작하지 않습니다. 일반적으로 None을 기입합니다.)
- PARTITION 컬럼의 값은 14자리 숫자(YYYYMMDDhhmmss)로 이루어져 있어야 합니다.
- 기본적으로 데이터는 PARTITION을 통해 나누고, 데이터 분산을 더 시키기 위해서 KEY 값을 사용합니다.
- KEY 값이 지나치게 늘어나거나, PARTITION RANGE 가 지나치게 작을 경우 분산된 Backend 수가 많아져서 성능 저하를 야기할 수 있습니다.

3 1.5 버전 신규 기능

3.1 FTS (Full-Text-Search)

FTS 기능을 사용하면 문자열 기반 고속 검색을 위해서 단어단위로 인덱스를 생성합니다. FTS 기능을 사용하려면 FTS 전용 SQL 문을 사용해야 합니다.

FTS 기능 사용 테이블 생성

```
CREATE virtual TABLE TEST_TABLE USING FTS4 (
   k         TEXT,
   p         TEXT,
   a         TEXT
)
datascope       LOCAL
ramexpire       30
diskexpire      1440
partitionkey    k
partitiondate   p
partitionrange  10
;
```

FTS 기능을 사용해서 데이터를 검색하는 쿼리

```
iplus> SELECT * FROM TEST_TABLE WHERE a MATCH '1';
Ret : +OK Success

 K             P             A
=======================================================
 k2            20110616000000 1
 k3            20110616000000 1.2
 k5            20110616000000 0.1
=======================================================
3 row in set
0.1774 sec
```

```
iplus> SELECT * FROM TEST_TABLE WHERE a MATCH '^1';
Ret : +OK Success

 K             P             A
=======================================================
 k2            20110616000000 1
 k3            20110616000000 1.2
=======================================================
2 row in set
0.0457 sec
```

3.2 암호화

- 암호화 스펙
  - 암호화 알고리즘 : AES-256
  - 블럭사이즈 : 32 bit
  - HASH 알고리즘 : SHA256
- DB파일 암호화 제공

IRIS 를 DB암호화버전으로 설치시 모든 데이터가 자동으로 암호화되서 저장됩니다. IRIS 를 통해서 조회하면 자동으로 복호화되서 조회됩니다.

- Column 암호화 제공

일부 데이터를 IRIS 를 통해서 조회하더라도 평문이 보이지 않도록 암호화 합니다. 비 DB암호호버전 IRIS 도 사용가능합니다. 데이터 로딩시 암호화할 Column을 선택하면 IRIS 내부에서 암호화해서 저장합니다.

ENCRYPT 암호화 방법은 SELECT QUERY 에서 DECRYPT 함수를 사용해서 원문을 볼 수 있습니다. 로딩시 컬럼이름 앞에 @ 을 붙여서 표시합니다.

```
    [iris@]$ cat LOCAL_TEST_TABLE.ctl
    k
p
@a

[iris@]$ cat LOCAL_TEST_TABLE.dat
k,20110616000000,b
k,20110616000000,c

[iris@]$ python 06.load.py
+OK SUCCESS. success count : 2

iplus> select k,p,a from LOCAL_TEST_TABLE;
Ret : +OK Success

 K              P              A
=======================================================
 k              20110616000000 xjYSHPzXiQm6T/fUP5X4NPnn7...
 k              20110616000000 gJjVSp9y6Qfnf16tO9XaHydq9...
=======================================================
2 row in set
0.1116 sec

iplus> select k,p,decrypt(a) from LOCAL_TEST_TABLE;
Ret : +OK Success

 K              P              DECRYPT(A)
=======================================================
 k              20110616000000 b
 k              20110616000000 c
=======================================================
2 row in set
0.0505 sec

iplus>
```

HASH 암호화 방법은 복호화가 불가능한 HASH 암호화를 적용합니다. 로딩시 컬럼이름 앞에 # 을 붙여서 표시합니다.

```
[iris@]$ cat LOCAL_TEST_TABLE.ctl
k
p
#a

[iris@]$ cat LOCAL_TEST_TABLE.dat
k,20110616000000,b
k,20110616000000,c

[iris@]$ python 06.load.py
+OK SUCCESS. success count : 2

iplus> select k,p,a from LOCAL_TEST_TABLE;
Ret : +OK Success

 K             P             A
=======================================================
 k             20110616000000 3e23e8160039594a33894f656...
 k             20110616000000 2e7d2c03a9507ae265ecf5b53...
=======================================================
2 row in set
0.1116 sec

iplus>
```

3.3 Hadoop EcoSystem 지원

IRIS 에 저장된 데이터를 Hadoop EcoSystem 을 통해서 검색할 수 있습니다. IRIS 에서 제공하는 EcoSystem 용 드라이버를 사용해서 IRIS 를 HDFS 인터페이스로 추상화해서 연결합니다. IRIS 로부터는 데이터를 읽을수만 있기 때문에 데이터 분석을 수행하기 위해 HDFS 를 사용하는 경우 (Hadoop 등) HDFS 가 필요합니다.

- 지원 EcoSystem

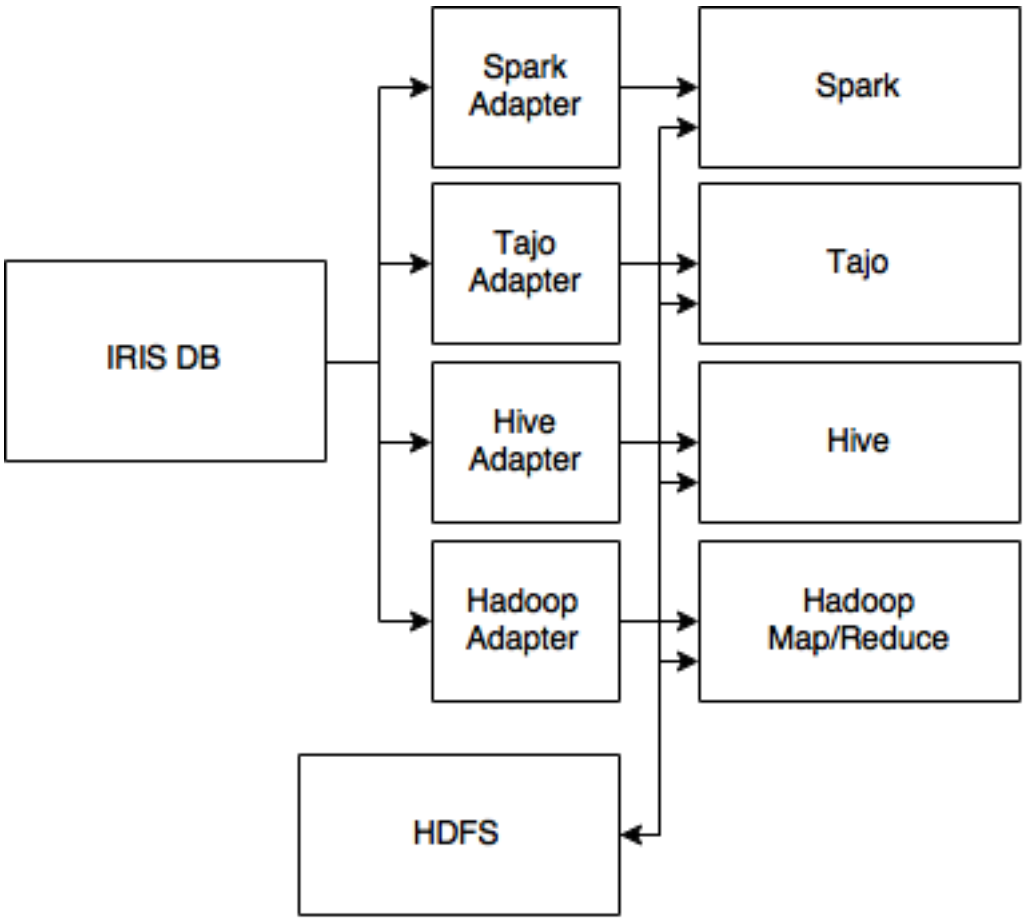

그림 10: Hadoop EcoSystem 구성도

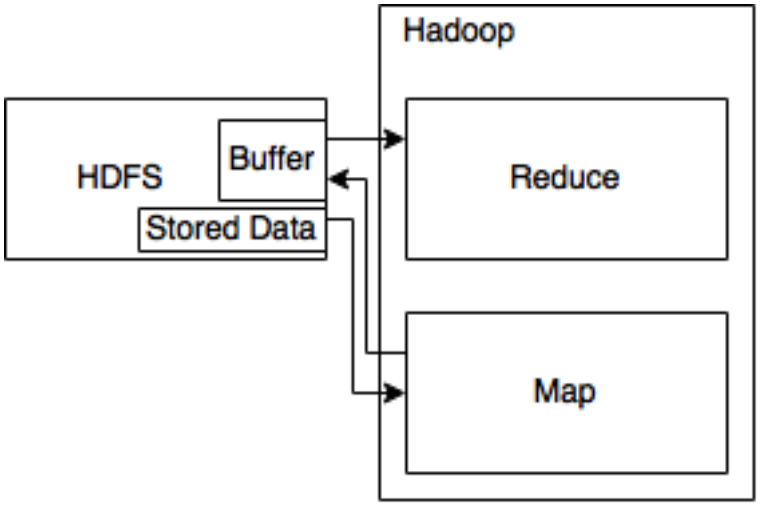

그림 11: 일반적인 HDFS 기반 Map/Reduce 구성도

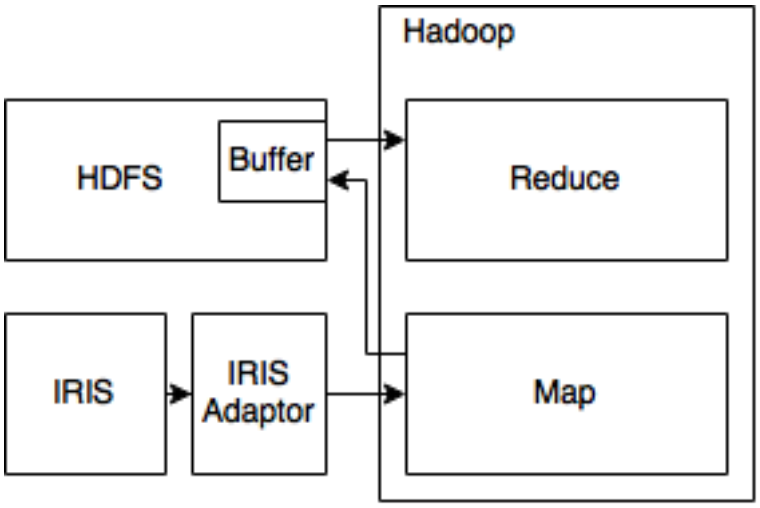

그림 12: IRIS 를 사용한 Map/Reduce 구성도

Spark

```
사용예시)
from pyspark import SparkContext
sc = SparkContext(appName="test")
from sparkiris import sparkiris
sc = sparkiris(sc)
sc.set_iris_info('192.168.0.170', 'test','test', 35210, 35110, \
    35000, 35678, 35800)
print sc.irisDB('select * from benchtest_100m;').count()
```

Tajo

```
사용예시)
default> SELECT ID, INT_KEY from BENCHTEST_100M WHERE key = 'k0'
    AND partition = '20141020092000' LIMIT 3;
Progress: 0%, response time: 0.432 sec
Progress: 0%, response time: 0.433 sec
Progress: 0%, response time: 0.835 sec
Progress: 0%, response time: 1.637 sec
Progress: 0%, response time: 2.639 sec
Progress: 100%, response time: 2.857 sec
id, int_key
--------------------
5,  50
6,  10
10,  50
(3 rows, 2.857 sec, 16 B selected)
default>
```

Hive

Hadoop Map/Reduce

3.4 시스템 정보 제공기능

SQL 명령어를 통해 시스템정보를 제공합니다. 옵션을 설정해서 기간별 누적정보를 제공할 수 있습니다. 자세한 내용 및 사용법은 아래의 CLI 목록에 있습니다.

```
iplus> .statistics system
Ret : +OK Success

 UPDATE_TIME   NODE_ID       NODE_IP       SYS_STATUS    ADM_STATUS    HOST_NAME
OS_NAME       OS_VERSION    OS_TYPE       NET_NAME      NET_TYPE      NET_MAC
NET_IN_PACKET NET_OUT_PACKET NET_IN_BYTE   NET_OUT_BYTE  CPU_CLOCK
CPU_CORE      CPU_USAGE     CPU_L_AVG     CPU_IOWAIT    RAM_TOTAL
RAM_USAGE_FILE RAM_USAGE_PROCESS RAM_SWAP_TOTAL RAM_SWAP_USAGE
HDD_TOTAL     HDD_USAGE
=======================================================
201506040845  0             192.168.131.13 VALID         ENABLE        ...
201506040845  1             192.168.131.131 VALID         ENABLE        ...
201506040845  2             192.168.131.132 VALID         ENABLE        ...
=======================================================
4 row in set
0.0206 sec

iplus>
```

3.5 Session 모니터링 기능

현재 실행중인 쿼리의 상태정보를 출력합니다. 조건에 맞는 세션을 조회하거나, 임의의 세션을 강제종료할 수 있습니다. 기본실행시 현재 실행중인 최신 10건의 세션 정보를 출력합니다. 자세한 내용 및 사용법은 아래의 CLI 목록에 있습니다.

```
iplus> .session list
Ret : +OK Success
```

```
SID           QUERY_STRING  START_TIME    END_TIME      TYPE          NODE
PID           RESULT
=======================================================
20150604140235_2_19577_32 select        20150604140235 20150604140235 END   ...
20150604140235_2_19577_33 node          20150604140235 20150604140235 END   ...
20150604140235_2_19577_34 select        20150604140235 20150604140235 END   ...
20150604140235_2_19577_35 system        20150604140235 20150604140235 END   ...
20150604140235_2_19577_36 select        20150604140235 20150604140236 END   ...
20150604140236_2_19577_37 system        20150604140236 20150604140236 END   ...
20150604140236_2_19577_38 select        20150604140236 20150604140236 END   ...
20150604140236_2_19577_39 system        20150604140236 20150604140236 END   ...
20150604140236_2_19577_40 select        20150604140236 20150604140236 END   ...
20150604140236_2_19577_41 system        20150604140236 20150604140236 END   ...
=======================================================
10 row in set
0.2966 sec
```

작업 종료 기능 제공

```
iplus> .session term 20150604140235_2_19577_34
Ret : +OK kill 20150604140235_2_19577_34 session.

6.2503 sec

iplus>
```

3.6 IRIS Client 제공

IRIS API 파일과 IPLUS (IRIS CLI Interface) 를 하나로 묶은 통합 패키지를 제공합니다. IPLUS 를 임의의 서버에 설치해서 원격지의 IRIS에 접속할 수 있습니다. IRIS 는 대부분의 작업을 CLI 명령어로 제공하므로 IRIS Client 만으로도 원격지의 IRIS 를 제어, 관리 할 수 있습니다. 자세한 내용 및 사용법은 아래의 CLI 목록에 있습니다.

3.7 IRIS 고도화

3.7.1 DTD

DTD는 기존 IRIS에서 데이터를 동기화, 삭제를 진행하기 위해서 사용하는 BTD, BSD 데몬은 대체하게 위해서 만들어진 데몬입니다. 기존의 BTD, BSD 에서 하는 모든 작업을 DTD에서 수행하며, 코드 최적화 및 범용성을 가지도록 하였습니다.

3.7.2 IRIS FileSystem

기존 IRIS의 슬레이브 노드에서는 각각의 데이터 위치를 Ram 파일 시스템의 링크를 참조하는 방식으로 진행되었습니다. 이는 불필요한 메모리 소모 및 관리 어려움을 야기하였습니다. 이를 해결하고자 IRIS FileSystem 을 도입하였습니다. IRIS FileSystem은 기존의 위치 정보를 따로 관리 함으로써 메모리 소모 및 관리의 편리성을 확보하였습니다.

3.7.3 PR / PL

기존 IRIS PM 에서 하는 일이 대부분 IRIS FileSystem 내부로 들어감에 따라 PM은 제거되고 데이터 expire 관리를 위해 PR / PL 이 개발되었습니다.

3.7.3.1 PL [Partition Locator] PL은 Partition Locator로 파티션 위치 정보를 관리하며, RAM -> SSD or DISK, SSD -> DISK 의 이동을 관리합니다.

3.7.3.2 PR [Partition Remover] PR은 Partition Remover로 파티션 및 기타 파일 삭제를 관리합니다. 삭제를 관리하는 항목은 아래와 같습니다.

- 디스크에서 삭제 대기중인 파티션
- 사용하지 않는 락 파일
- 임시 파일