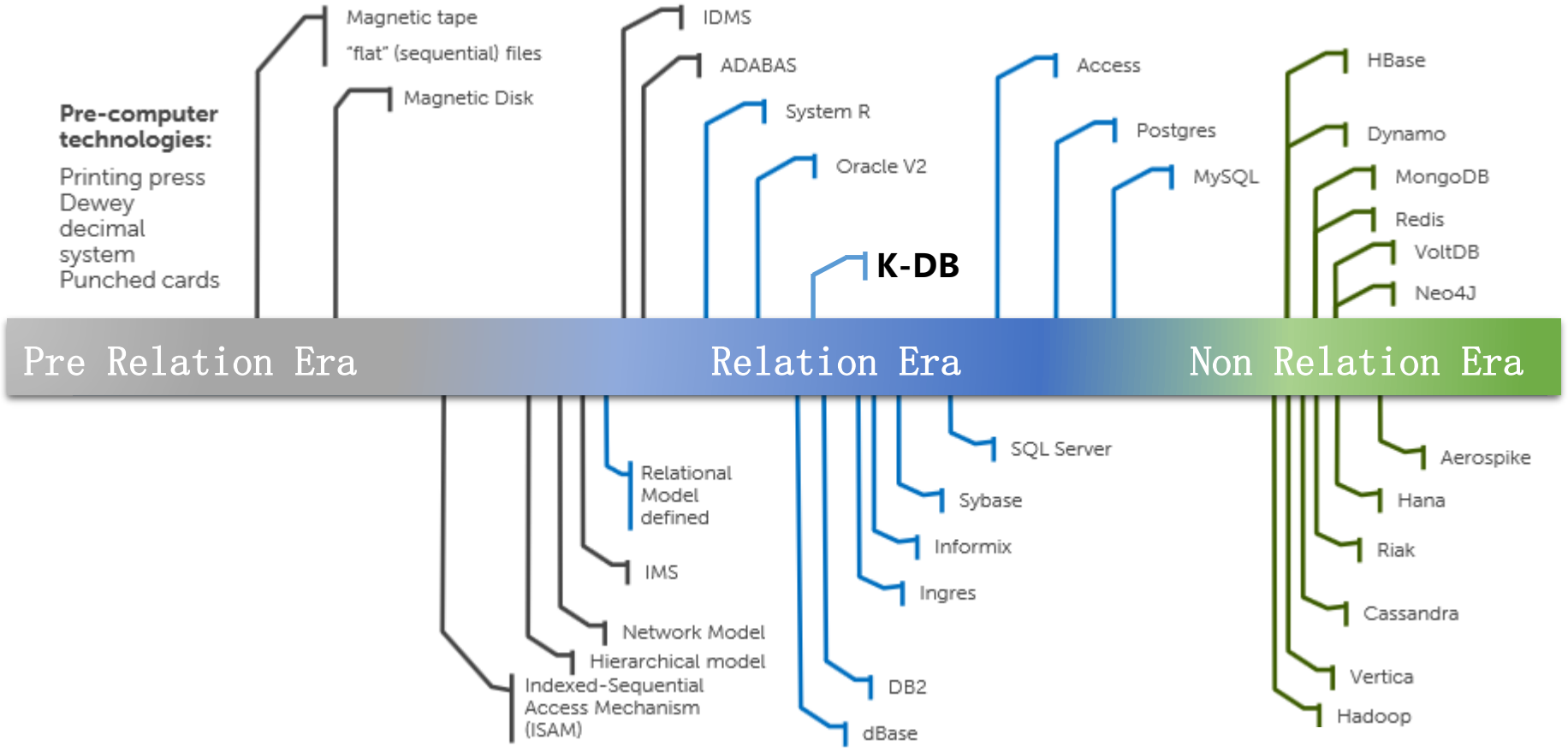
inspur 浪潮

传统数据库运维及面临的挑战

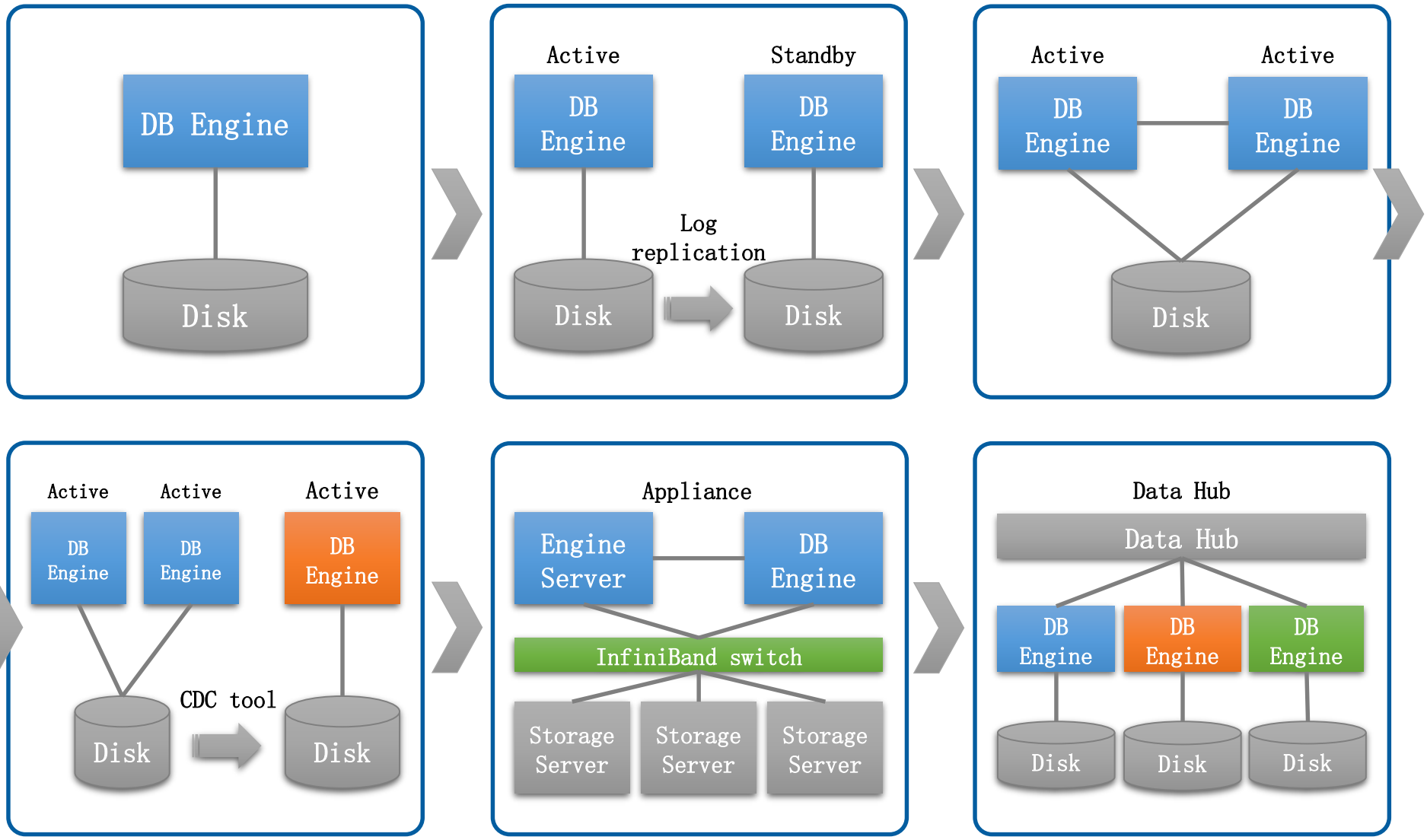
浪潮 架构师

金学东

数据库技术发展


Pre-computer technologies:
Printing press
Dewey decimal system
Punched cards
Magnetic tape
"flat" (sequential) files
Magnetic Disk
IDMS
ADABAS
System R
Oracle V2
K-DB
Access
Postgres
MySQL
HBase
Dynamo
MongoDB
Redis
VoltDB
Neo4J
Pre Relation Era
Relation Era
Non Relation Era
Relational Model defined
IMS
Network Model
Hierarchical model
Indexed-Sequential Access Mechanism (ISAM)
SQL Server
Sybase
Informix
Ingres
DB2
dBase
Aerospike
Hana
Riak
Cassandra
Vertica
Hadoop

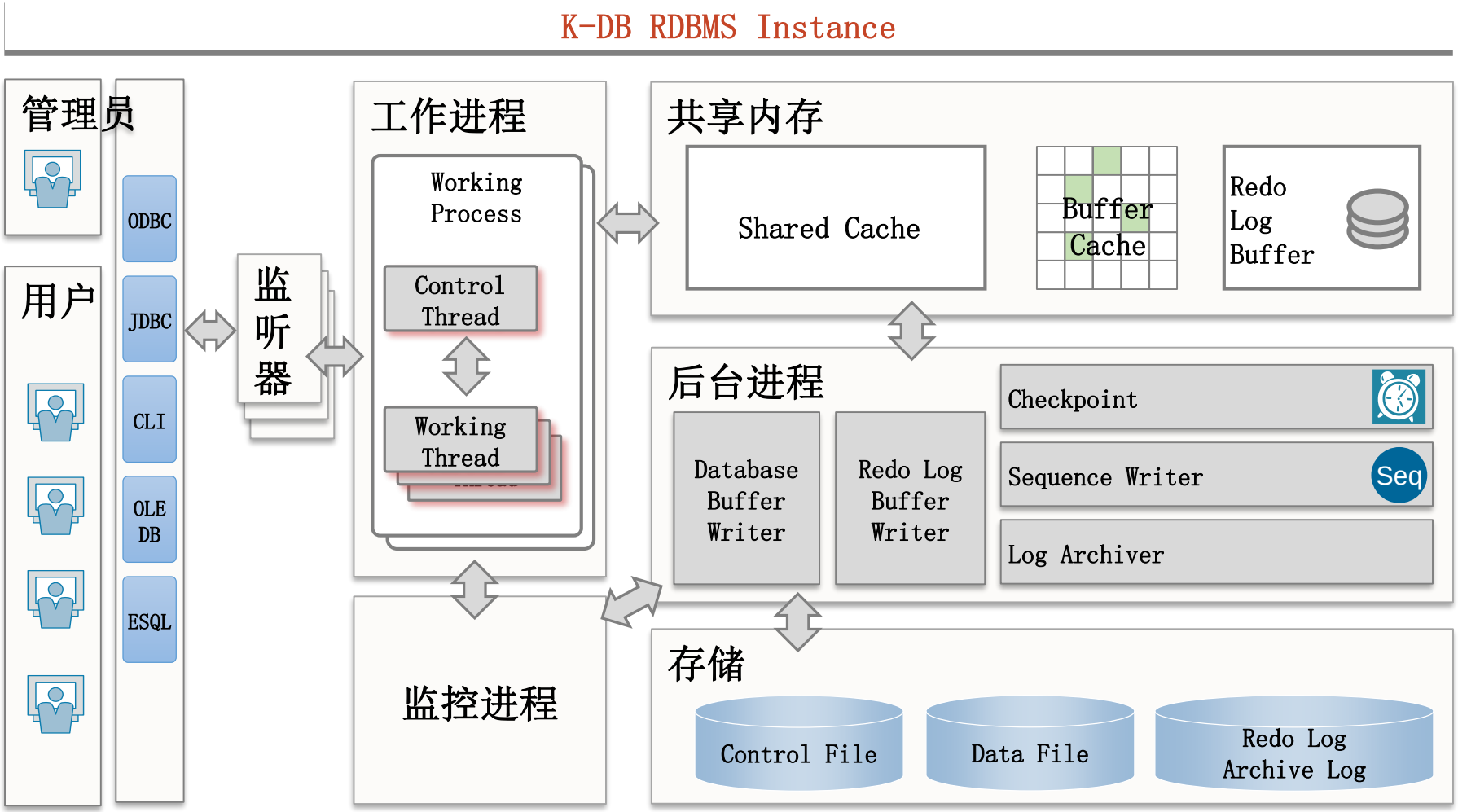
我们的故事


DB Engine
Disk
Active
Standby
DB Engine
DB Engine
Log replication
Disk
Disk
Active
Active
DB Engine
DB Engine
Disk
Active
Active
Active
DB Engine
DB Engine
DB Engine
CDC tool
Disk
Disk
Appliance
Engine Server
DB Engine
InfiniBand switch
Storage Server
Storage Server
Storage Server
Data Hub
Data Hub
DB Engine
DB Engine
DB Engine
Disk
Disk
Disk

雏形

架构图

- K-DB数据库一开始便在工作进程上采用了多进程多线程架构，但是在后台进程、日志、数据文件管理方式上还是更多地遵循了知名数据库的成熟架构。



监控方法 － 日志跟踪

- 早期的 K-DB数据库并不具备足够多的监控事件及统计工具。初期的数据库监控及问题分析，基本依靠数据库日志及Trace日志进行分析并解决。

DBMS log

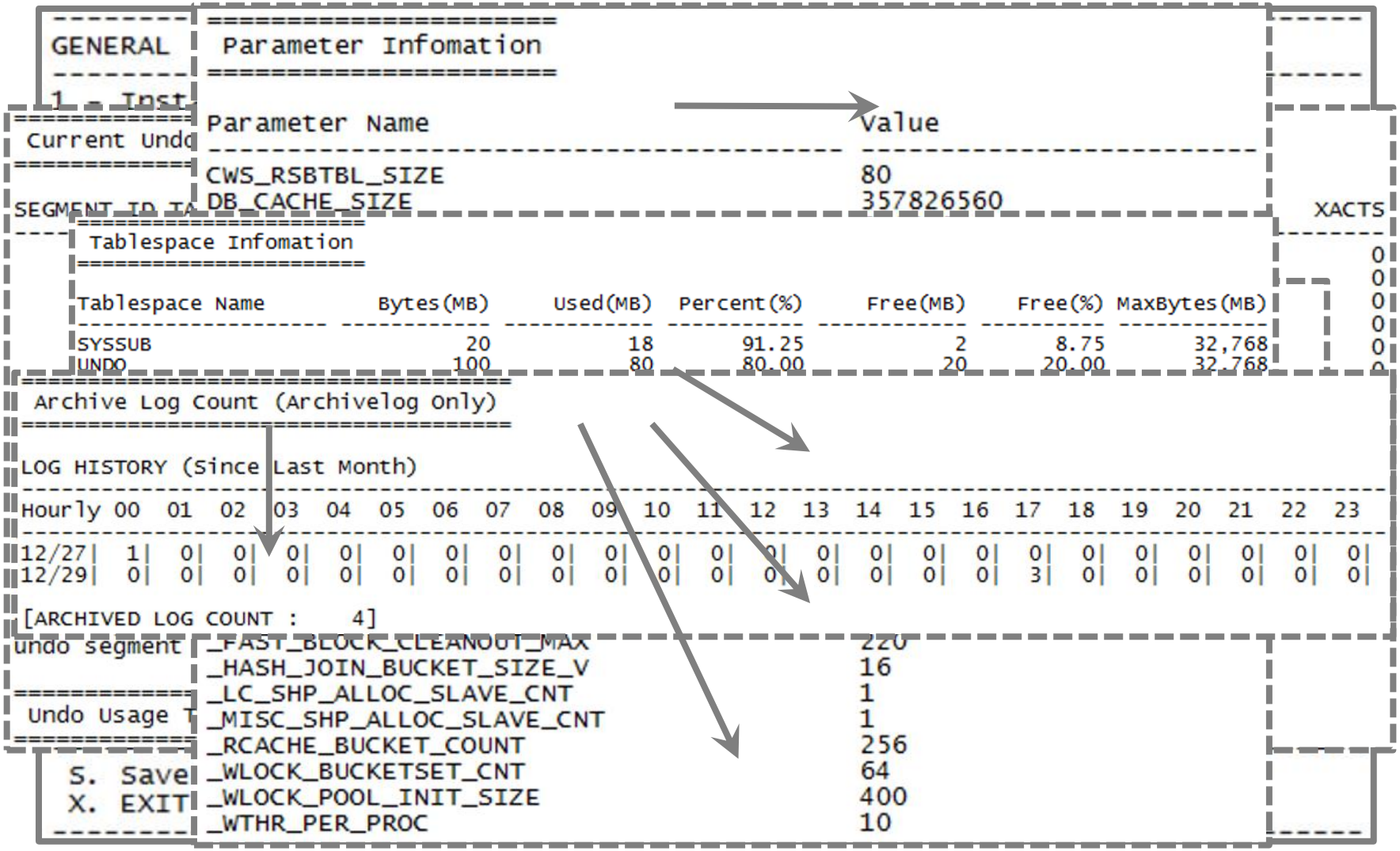
```
---------------------------------------------------------------------------
K-DB started up. protocol version 2.8

MTHR(tbsvr) is ready.     proc_idx(tid) :0, ospid :8338
CTHR(tbsvr_WT001) is ready.   proc_idx(tid) :1, ospid :8349
CTHR(tbsvr_WT002) is ready.   proc_idx(tid) :2, ospid :8350
CTHR(tbsvr_WT003) is ready.   proc_idx(tid) :3, ospid :8351
AGENT(tbsvr_SEQW) is ready.   proc_idx(tid) :4, ospid :8352
LGWR(tbsvr_LGWR) is ready.  proc_idx(tid) :5, ospid :8353
LARC(tbsvr_LOGA) is ready.  proc_idx(tid) :6, ospid :8354
CKPT(tbsvr_CKPT) is ready.  proc_idx(tid) :7, ospid :8355
DBWR(tbsvr_BLK001) is ready.  proc_idx(tid) :8, ospid :8356
DBWR(tbsvr_BLK002) is ready.  proc_idx(tid) :9, ospid :8357

DB NAME = kdb
CONTROL FILE = /home/kdb/database/kdb/c1.ctl
BLOCK SIZE = 8192
NLS CHARACTER SET = UTF8
---------------------------------------------------------------------------
2015/09/07 22:18:55.872 [DDL][0]  20 Executing DDL: create user test identified by ****
2015/09/07 22:19:06.088 [DDL][0]  20 Executing DDL: grant dba to test
2015/09/07 22:20:08.727 [DDL][0]  20 Executing DDL: CREATE TABLE EMPLOYEE (
    EMP_NO          VARCHAR(8)      NOT NULL,    /* ?? ID */
    EMP_NAME        VARCHAR(20)     ,            /* ??      */
    HIREDATE        DATE            ,            /* ????    */
    SALARY          NUMBER(8,3)     ,            /* ??      */
    BONUS           NUMBER(8,3)     ,            /* ??    */
    DEPT_CD         VARCHAR(4)      ,            /* ????  */
    MANAGER       VARCHAR(8)     ,            /* ?? */
    CONSTRAINT EMPLOYEE_PK
       PRIMARY KEY (EMP_NO)
       USING INDEX
       PCTFREE       20
       TABLESPACE    USERS
)
    PCTFREE    10
    TABLESPACE USERS
2015/09/07 22:23:32.917 [COM][0]  21 Internal Error with condition 'cvi->cv->seqno != TCBLK_SEQNO_APP =
2015/09/07 23:26:02.603 [CLD][0]   4 slow DB BLOCK I/O (  261ms per block) at 2015/09/07 23:16:24.808
2015/09/08 00:06:07.058 [CLD][0]   4 slow LOG BLOCK I/O (   65ms per block) at 2015/09/08 00:01:15.545
2015/09/08 00:06:07.058 [CLD][0]   4 slow DB BLOCK I/O (  228ms per block) at 2015/09/08 00:01:40.665
2015/09/09 00:07:47.363 [CLD][0]   4 slow LOG BLOCK I/O (   60ms per block) at 2015/09/09 00:01:10.287
2015/09/09 10:32:09.768 [FRM][0]  11 get shutdown request msg. mode=1
2015/09/09 10:32:09.838 [CLK][0]   7 CHECKPOINT done     3ms,    6 blocks at 0000.000c7b7c/3.335759.26.
2015/09/09 10:32:09.845 [CLK][0]   7 CHECKPOINT done     0ms,    4 blocks at 0000.000c7b7c/3.335759.26.
2015/09/09 10:32:09.845 [CLK][0]   7 CKPT write disabled
2015/09/09 10:32:09.858 [CLC][0]   7 LOG FILE CLOSE: thread=0, current log=3, rba=3.335758
2015/09/09 10:32:09.858 [CLC][0]   7 REDO THREAD #0 CLOSED: 0000.000c7b7c(0000.000c7b7c), 0 opens
2015/09/09 10:32:09.858 [CLC][0]   7 DB CLOSE: C 0000.000c7b7c, $ 0000.000c7b7c, L 3.335758/0000.000c7b
2015/09/09 10:32:09.989 [FRM][0]   0 server shut down.
2015/09/09 10:32:56.953 [FRM][0]   0 show tip [kdb.tip]
---------------------------------------------------------------------------
# tip file generated from /home/kdb/config/tip.template (Thu Aug  6 16:58:04 CST 2015)
#---------------------------------------------------------------------------
```

Trace log

SET AUTOT[RACE] {OFF|ON|TRACE[ONLY]}
[EXP[LAIN]] [STAT[ISTICS]]

```
BEGIN
    DBMS_OUTPUT.ENABLE;

  FOR PSTSR50T_CUSRSOR_REC IN PSTSR50T_CUSRSOR LOOP

    V_REQ_NO := PSTSR50T_CUSRSOR_REC.REQ_NO;
    V_ASSIGN_SEQ := PSTSR50T_CUSRSOR_REC.ASSIGN_SEQ;
    V_SERVE_SEQ := PSTSR50T_CUSRSOR_REC.SERVE_SEQ;
    V_PROD_SEQ := PSTSR50T_CUSRSOR_REC.PROD_SEQ;
    V_LICENSE_KIND := PSTSR50T_CUSRSOR_REC.LICENSE_KIND;
    V_LICENSE_CLS := PSTSR50T_CUSRSOR_REC.LICENSE_CLS;


    CURSOR_COUNT := PSTSR50T_CUSRSOR%ROWCOUNT;

    DBMS_OUTPUT.PUT_LINE('==== 鐢ㄦ埛?PSTSR50T COUNT:===' || CURSOR_COUNT);
    DBMS_OUTPUT.PUT_LINE('REQ_NO       :' || V_REQ_NO);
    DBMS_OUTPUT.PUT_LINE('ASSIGN_SEQ   :' || V_ASSIGN_SEQ);
    DBMS_OUTPUT.PUT_LINE('SERVER_SEQ   :' || V_SERVE_SEQ);
    DBMS_OUTPUT.PUT_LINE('PROD_SEQ     :' || V_PROD_SEQ);
    DBMS_OUTPUT.PUT_LINE('LICENSE_KIND :' || V_LICENSE_KIND);
    DBMS_OUTPUT.PUT_LINE('LICENSE_CLS  :' || V_LICENSE_CLS);

--PINST10T 鏁版嵁鍦ㄦ彃鍏ヨ鍙栨椂闂存€ 20061024 ` 20061104 鍜?鏁呴殰 鐢ㄦ埛鍙樻
--LICENSEKIND 鍜€ LICENSECLS 瀛?鏁版   涓嶅尮閰?UPDATE 澶勭悊.

    UPDATE PSTSR50T
       SET LICENSE_KIND = V_LICENSE_CLS,
           LICENSE_CLS = V_LICENSE_KIND
    WHERE REQ_NO = V_REQ_NO
      AND ASSIGN_SEQ = V_ASSIGN_SEQ
      AND SERVE_SEQ = V_SERVE_SEQ
      AND PROD_SEQ =V_PROD_SEQ;


    DBMS_OUTPUT.PUT_LINE('UPDATE SUCCESS:' || CURSOR_COUNT);

  END LOOP;

   DBMS_OUTPUT.PUT_LINE('鏇存柊 鐢ㄦ埛?鏉?' || CURSOR_COUNT);
END UPDATE_PSTSR50T_LICENSE;

08/06 17:11:01.107756 [DDL][0] 20 ddl_fram:207 DDL execution succeeded
08/06 17:11:01.109107 [DDL][0] 20 ddl_fram:185 Executing DDL: CREATE OR REPLACE PROCEDURE UPDATE_RECRUIT_STATE
(
  P_REQUEST_NO  RCSCHOOT.REQUEST_NO%TYPE,
  P_REQUEST_SEQ RCSCHOOT.REQUEST_SEQ%TYPE,
  P_RESULT     OUT VARCHAR2
)
AS
  V_REQUEST_NO  RCSCHOOT.REQUEST_NO%TYPE  := P_REQUEST_NO;
```

监控方法 - kdBinary



GENERAL

1 . Inst

Current Undo

SEGMENT_ID IA

XACTS

Undo segment

Undo Usage T

S. Save

X. EXIT

Parameter Infomation

Parameter Name	Value
CWS_RSBTBL_SIZE	80
DB_CACHE_SIZE	357826560
_FAST_BLOCK_CLEANOUT_MAX	220
_HASH_JOIN_BUCKET_SIZE_V	16
_LC_SHP_ALLOC_SLAVE_CNT	1
_MISC_SHP_ALLOC_SLAVE_CNT	1
_RCACHE_BUCKET_COUNT	256
_WLOCK_BUCKETSET_CNT	64
_WLOCK_POOL_INIT_SIZE	400
_WTHR_PER_PROC	10

Tablespace Infomation

Tablespace Name	Bytes(MB)	Used(MB)	Percent(%)	Free(MB)	Free(%)	MaxBytes(MB)
SYSSUB	20	18	91.25	2	8.75	32,768
UNDO	100	80	80.00	20	20.00	32,768

Archive Log Count (Archivelog Only)

LOG HISTORY (Since Last Month)

Hourly	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
12/27	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12/29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0

[ARCHIVED LOG COUNT : 4]

备份/恢复

- K-DB数据库在早期，随支持逻辑备份及物理备份两种模式，但是在最重要的物理备份领域，只支持简单的冷备及基于操作系统命令实现的增量备份。

逻辑备份

```
$ kdexport/kdimport username=testuser
password=testpassword sid=test file=backup.dat full=y
$ kdexport/kdimport cfgfile=backup.cfg
```

表模式

Table definitions
Table data
Owner grants
Owner indexes
Table constraints
Table triggers

用户模式

Table definitions
Table data
Owner grants
Owner indexes
Table constraints
Table triggers
Views
Private synonyms
Sequences
PSM

全库模式

Table definitions
Table data
Owner grants
Owner indexes
Table constraints
Table triggers
Views
Private synonyms
Sequences
PSM
Roles
All synonyms
… …

物理备份

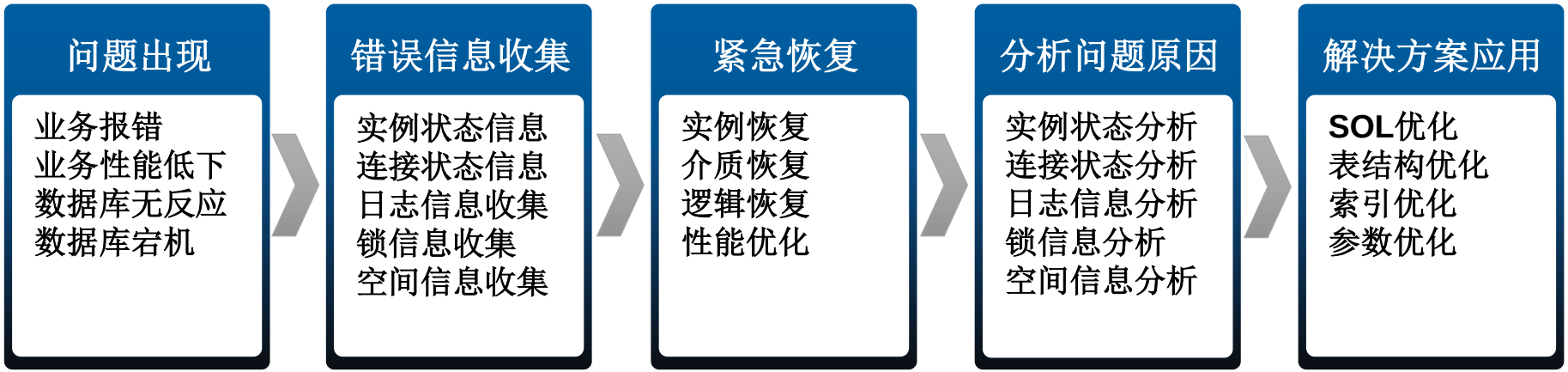
- 停机备份
  - 硬拷贝数据库相关文件
- 在线备份
  - 数据库运行状态下备份

```
SQL> select f.name
2 from v$tablespace t join v$datafile f on t.ts# = f.ts#
3 where t.name = 'USER';

NAME
------------------------------------------------------------
/disk3/user001.dbf

1 selected

SQL> alter tablespace SYSTEM begin backup;

altered

SQL> !cp /disk3/user001.dbf /backup/
SQL> alter tablespace  SYSTEM end backup;

altered

SQL>
```

运维方案

正常运维

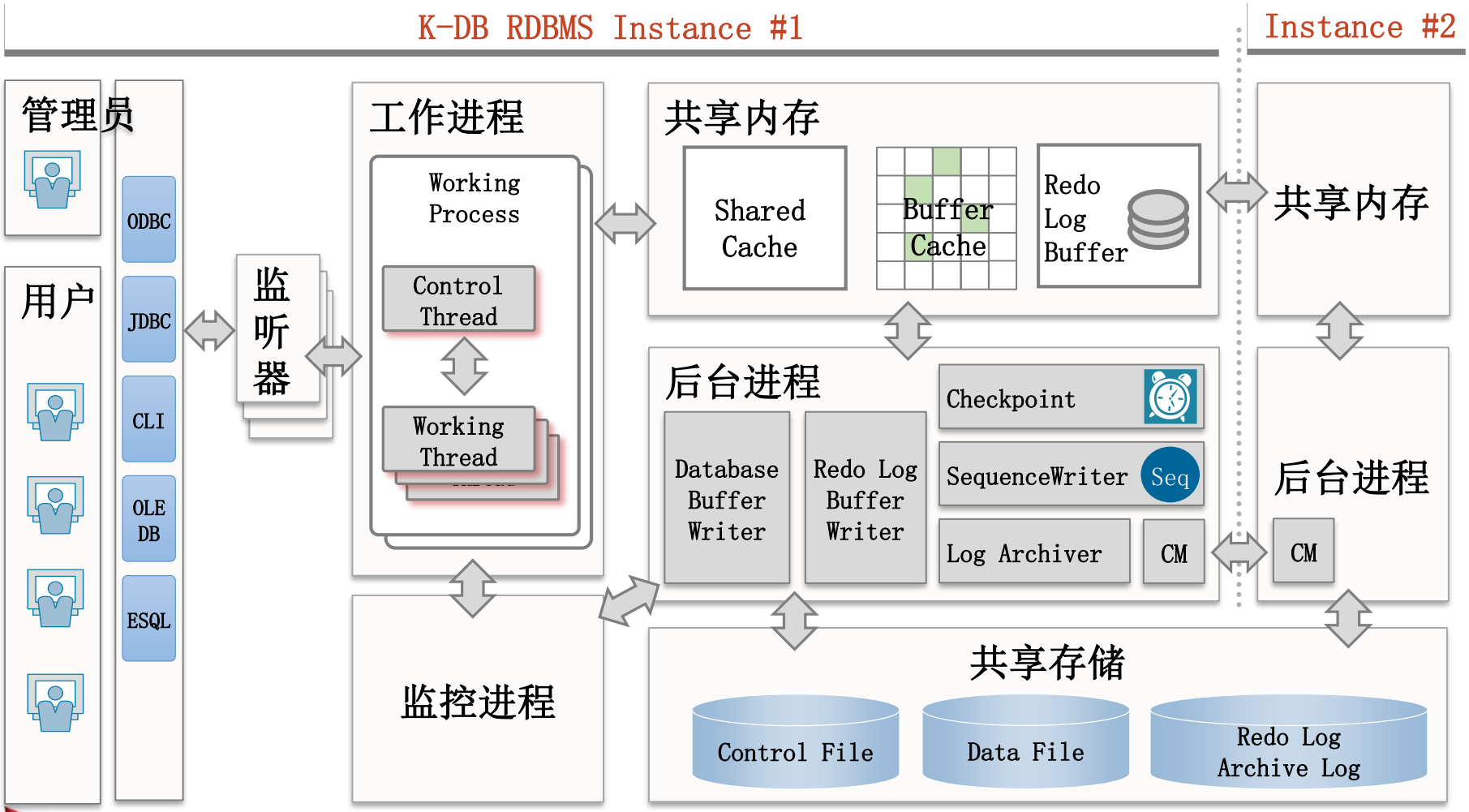
紧急运维



高可用

架构图

- K-DB早期版本还未来得及站稳市场，基于 RAC架构的高可用架构早已成为行业标准，日以继夜的追赶之下，K-DB高可用技术面世并逐渐成熟。

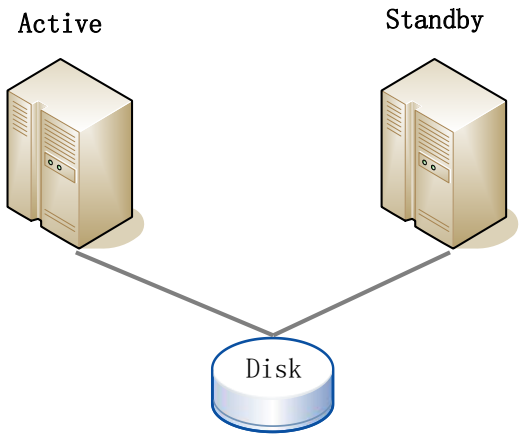


架构升级 - Active-Standby

- K-DB首先实现了基于共享存储的 H/A切换及基于日志同步的容灾架构。尤其，实现了容灾模式下的 Standby只读功能后，最后的难关便只剩下 RAC架构的实现。

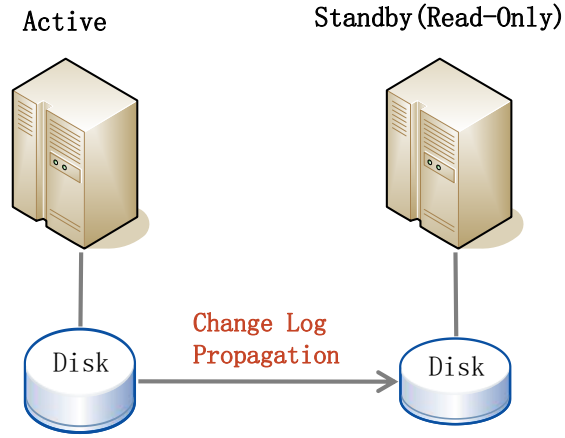
Active-Standby(Shared)

- 基于共享磁盘的 Active-Standby架构
- 同时有且只有一个节点处于运行状态
- 基于第三方HA软件进行切换



Active-Standby(Replication)

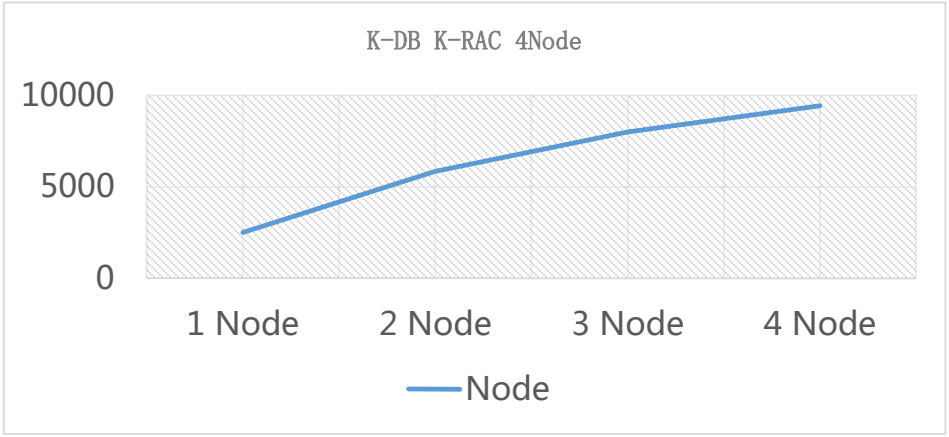
- 基于日志同步的 Active-Standby架构
- 同时有且只有一个节点以主节点模式运行，其他节点以 Standby模式（必要时可读）运行
- 基于日志同步插件（类似于 DataGuard）实现数据同步

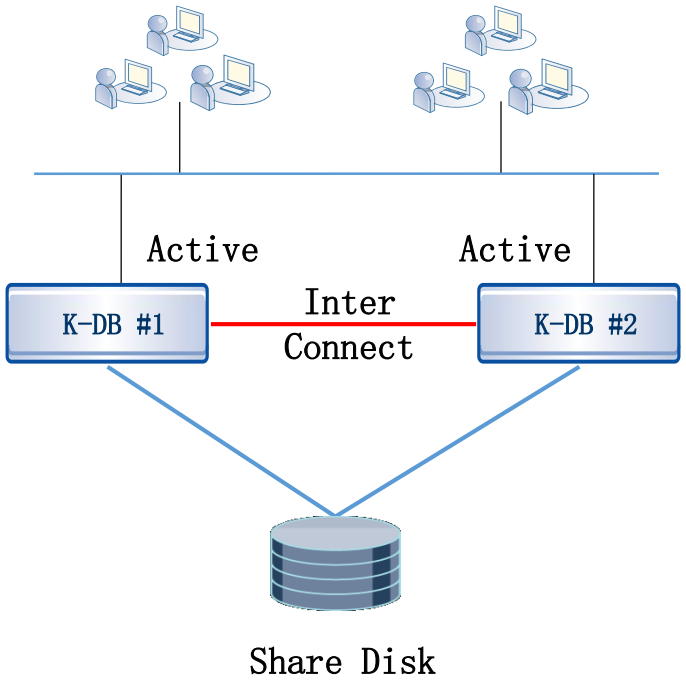


架构升级 – RAC

- K-DB首先实现了基于共享存储的 H/A切换及基于日志同步的容灾架构。尤其，实现了容灾模式下的 Standby只读功能后，最后的难关便只剩下 RAC架构的实现。

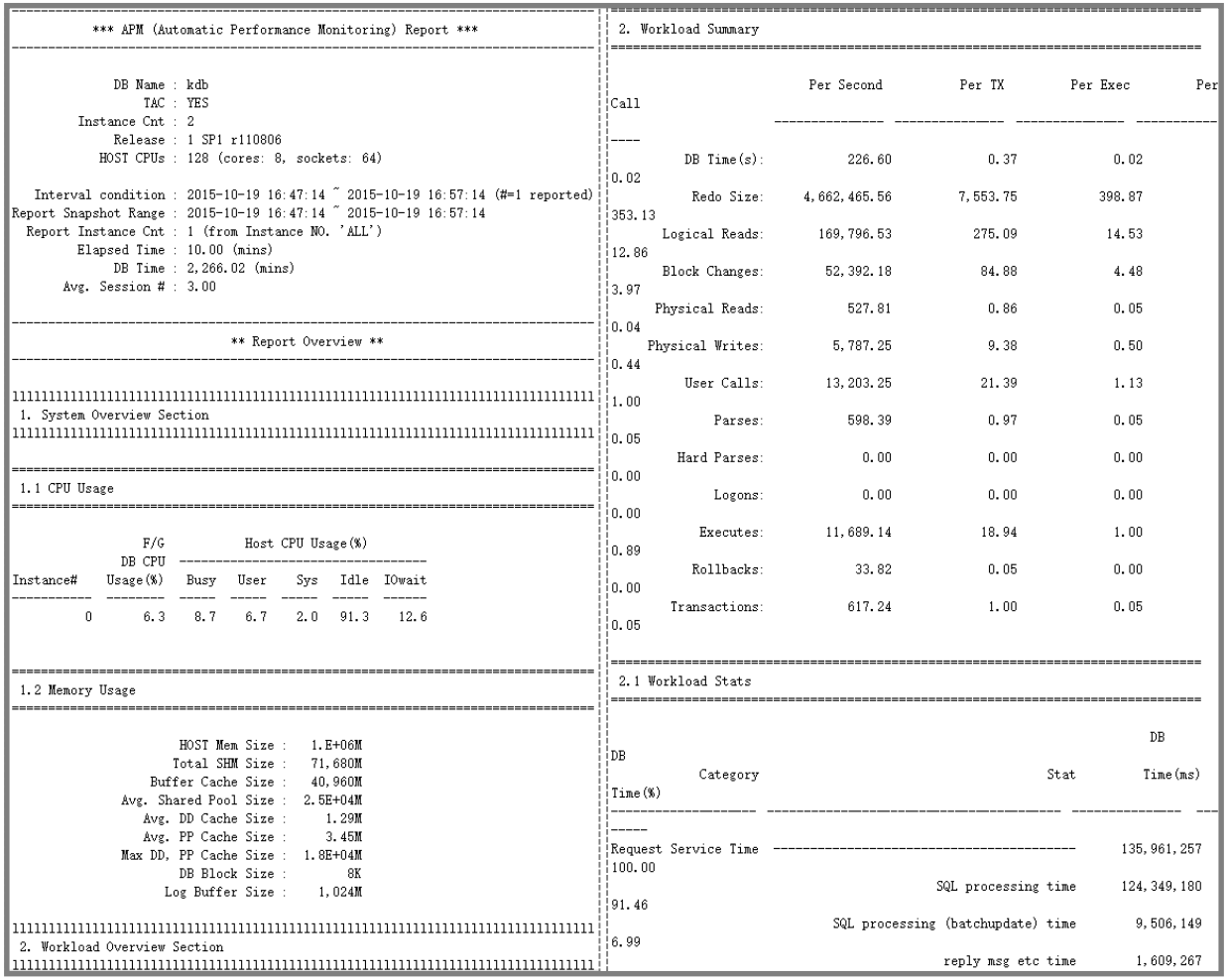
- DB 和系统发生故障时保证运营系统无中断运营
- 特定节点发生故障时快速恢复到正常节点
- 实现 Global Cache将 Disk I/O最小化
- 水平的添加Active节点，实现平滑扩容
- 局限：扩容节点数量存在限制





监控方法 - APM报告

- 基于架构的升级及改造，大大提升了K-DB的监控能力。基于快照实现的统计报告功能推出，大大减轻了分析压力（早期为 TXT版本）。



```
         *** APM (Automatic Performance Monitoring) Report ***

            DB Name : kdb
                TAC : YES
       Instance Cnt : 2
            Release : 1 SP1 r110806
          HOST CPUs : 128 (cores: 8, sockets: 64)

   Interval condition : 2015-10-19 16:47:14 ~ 2015-10-19 16:57:14 (#=1 reported)
Report Snapshot Range : 2015-10-19 16:47:14 ~ 2015-10-19 16:57:14
  Report Instance Cnt : 1 (from Instance NO. 'ALL')
         Elapsed Time : 10.00 (mins)
              DB Time : 2,266.02 (mins)
      Avg. Session # : 3.00

                              ** Report Overview **

 1. System Overview Section

 1.1 CPU Usage

                F/G           Host CPU Usage(%)
               DB CPU  ------------------------------
Instance#    Usage(%)   Busy   User    Sys   Idle  IOwait
----------  --------  ------  -----  -----  -----  ------
         0       6.3    8.7    6.7    2.0   91.3    12.6

 1.2 Memory Usage

                     HOST Mem Size :   1.E+06M
                     Total SHM Size :   71,680M
                  Buffer Cache Size :   40,960M
                Avg. Shared Pool Size :  2.5E+04M
                   Avg. DD Cache Size :    1.29M
                   Avg. PP Cache Size :    3.45M
                Max DD, PP Cache Size :  1.8E+04M
                        DB Block Size :       8K
                    Log Buffer Size :   1,024M

 2. Workload Overview Section

 2. Workload Summary

                              Per Second         Per TX         Per Exec         Per
Call                  ---------------  ---------------  ---------------  ----------
----
              DB Time(s):          226.60             0.37             0.02
0.02
               Redo Size:    4,662,465.56         7,553.75           398.87
353.13
            Logical Reads:      169,796.53           275.09            14.53
12.86
          Block Changes:       52,392.18            84.88             4.48
3.97
         Physical Reads:          527.81             0.86             0.05
0.04
        Physical Writes:        5,787.25             9.38             0.50
0.44
             User Calls:       13,203.25            21.39             1.13
1.00
                 Parses:          598.39             0.97             0.05
0.05
           Hard Parses:            0.00             0.00             0.00
0.00
                 Logons:           0.00             0.00             0.00
0.00
               Executes:       11,689.14            18.94             1.00
0.89
              Rollbacks:           33.82             0.05             0.00
0.00
            Transactions:          617.24             1.00             0.05
0.05

 2.1 Workload Stats

                                                                          DB
DB
          Category                                Stat                 Time(ms)
Time(%)
------------------  --------------------------------------  ---------------  ---
-----
Request Service Time  ----------------------------------------      135,961,257
100.00
                                    SQL processing time       124,349,180
91.46
                          SQL processing (batchupdate) time        9,506,149
6.99
                                    reply msg etc time        1,609,267
```

K-APM

基本信息查询

- CPU、内存使用率
- 负载状态显示
- 集群状态显示
- TOP 5 等待事件
- I/O运行状态
- PGA状态
- TOP N SQL

详细信息查询

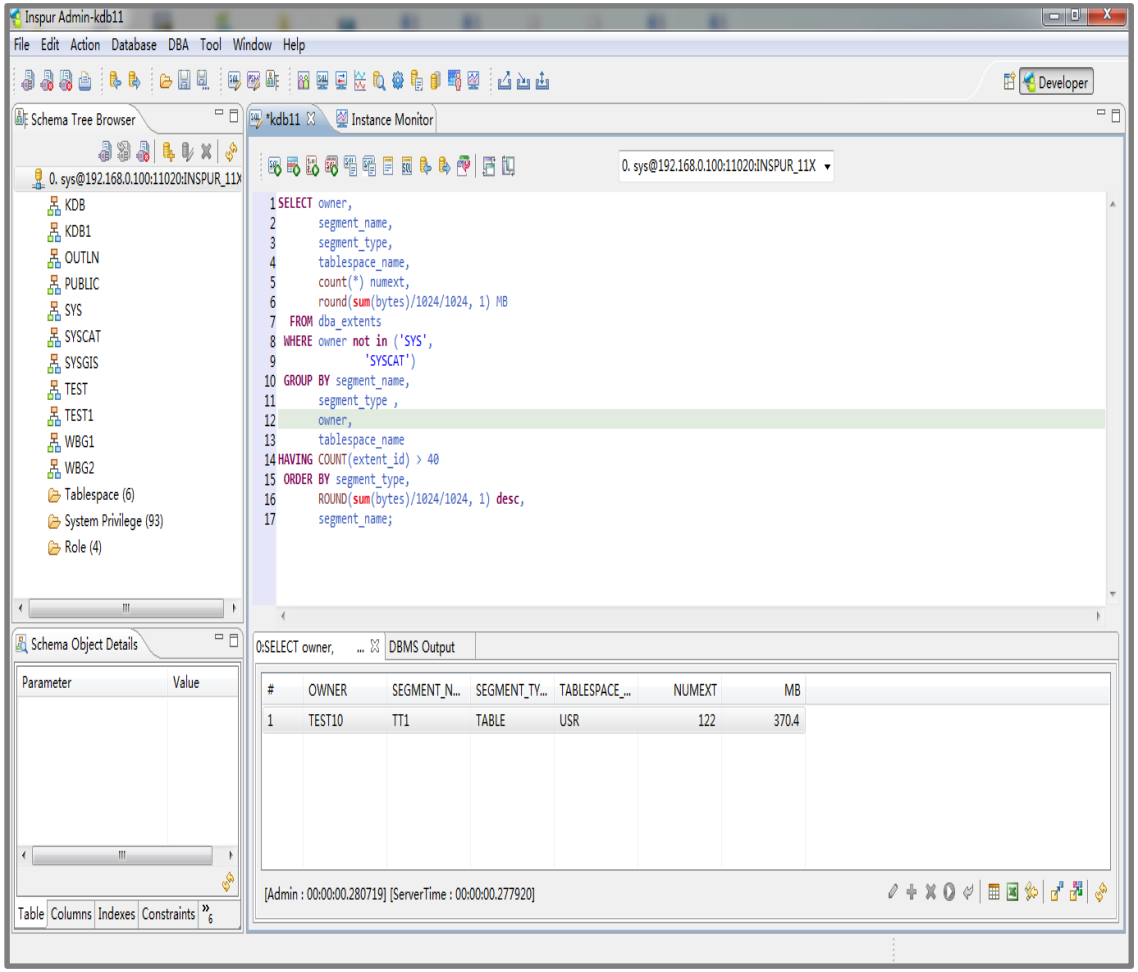
- OS 状态
- 共享池及各种缓存状态
- 等待事件详细描述
- 锁状态
- TOP N SQL及执行计划

其他

- 数据库运行参数

操作管理 - kdAdmin

- 作为数据库开发与管理的便利性考虑，第一款图形化数据库开发工具 kdAdmin面试。随着产品认知度的提高，第三方产品的适配也逐渐开始。



kdAdmin功能

- 开发
  - 用户及对象信息查询
  - SQL语句编辑与执行
  - 存储过程编辑与执行
- 监控
  - 会话监控及管理
  - 实例监控
  - 事务监视
- 管理
  - 参数管理
  - 表空间管理
  - 数据库导入与导出
- 分析及优化
  - 分析管理器
  - 图形化统计报告

备份/恢复 - RMGR & Flashback

- RMGR - 数据库自动化备份与恢复工具。我们对其满怀期望，但是开发初期，只有简单的在线全备与增量备份功能。闪回，也只能实现对于执行语句的回退。

RMGR备份恢复工具

- RMGR 基本功能
  - Online Full Backup
  - Incremental Backup
  - Automatic Recovery

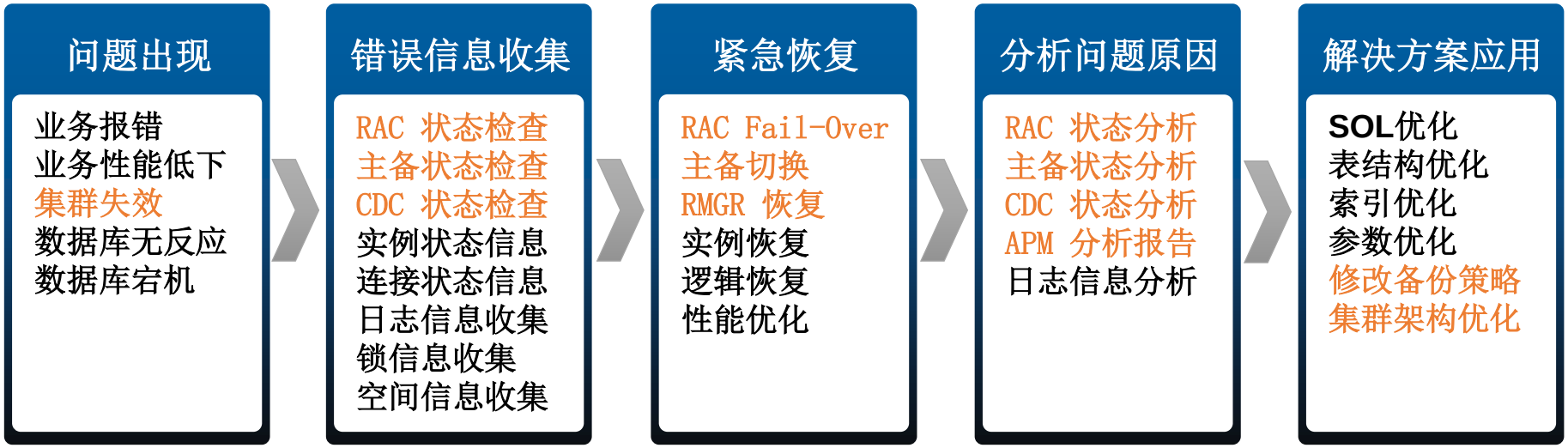
```
$ kdrmgr backup -i -o /home/winds/backup2 -b /home/winds/backup/rmgr.inf -v
============================================================
= Recovery Manager(RMGR) starts                            =
=                                                          =
= Inspur Corporation Copyright (c) 2008-. All rights reserved. =
============================================================
==========================================
  RMGR - INCREMENTAL backup
==========================================
DB connected
DBNAME: kdb
```

闪回功能

Flashback Query

运维方案

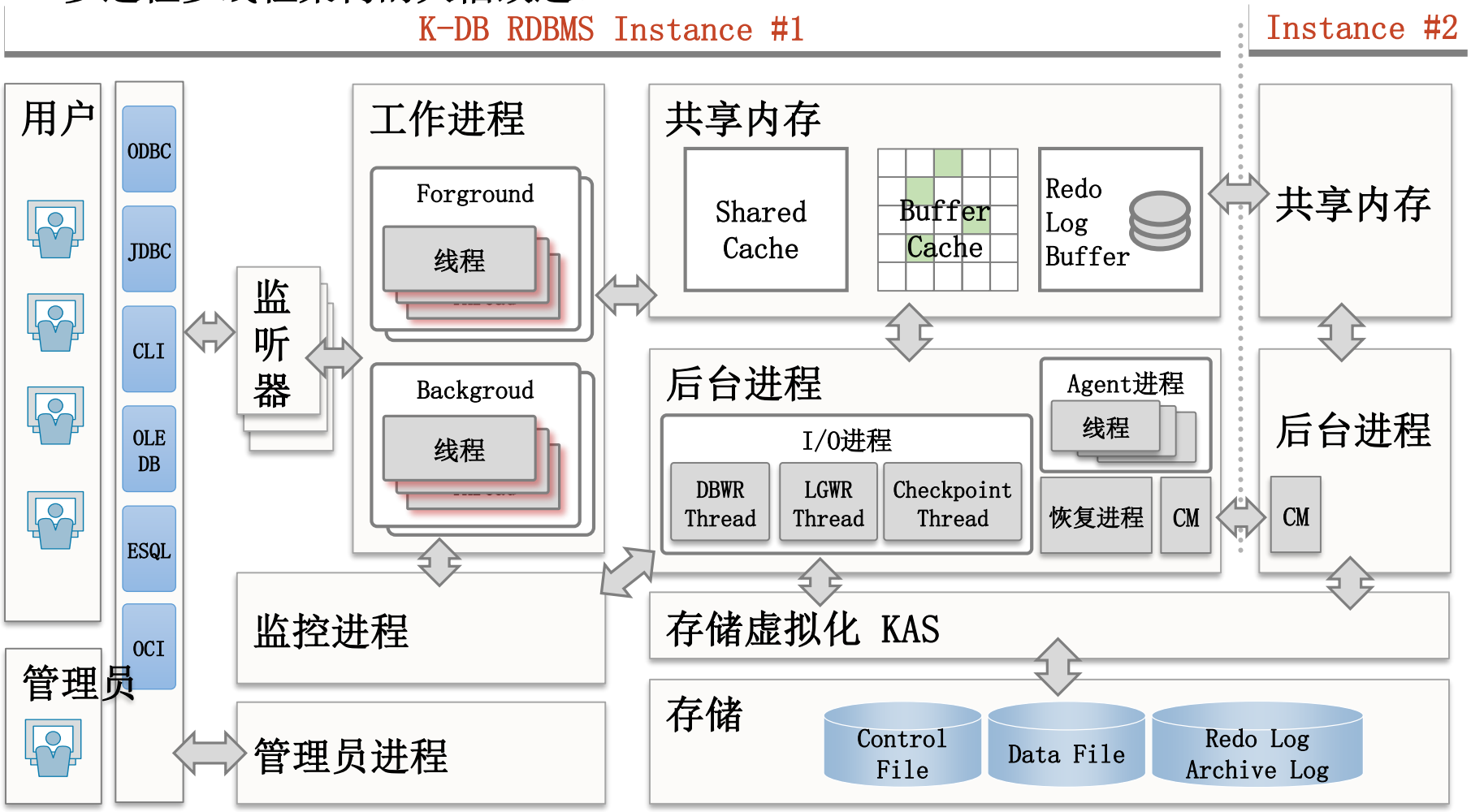

状态检查
RAC 状态检查
主备状态检查
CDC 状态检查
实例状态检查
连接状态检查
日志信息检查
锁信息检查
空间信息检查
系统维护
参数修改
重做日志维护
表空间维护
系统备份
逻辑备份
Export
RMGR 备份
全库冷备
在线备份
维护报告
运维日报
运维周报
运维月报


问题出现
业务报错
业务性能低下
集群失效
数据库无反应
数据库宕机
错误信息收集
RAC 状态检查
主备状态检查
CDC 状态检查
实例状态信息
连接状态信息
日志信息收集
锁信息收集
空间信息收集
紧急恢复
RAC Fail-Over
主备切换
RMGR 恢复
实例恢复
逻辑恢复
性能优化
分析问题原因
RAC 状态分析
主备状态分析
CDC 状态分析
APM 分析报告
日志信息分析
解决方案应用
SQL优化
表结构优化
索引优化
参数优化
修改备份策略
集群架构优化

現在

架构图

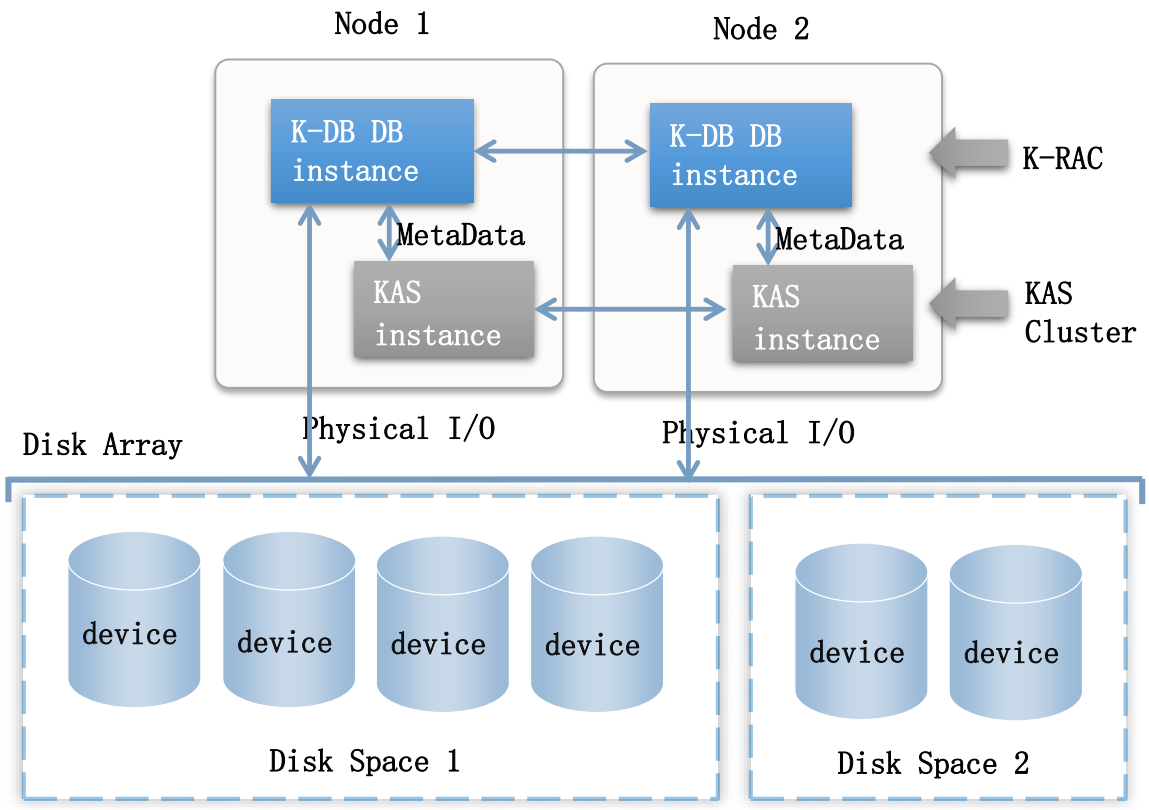
- K-DB 成熟版本的体系架构图，增加了KAS存储虚拟化、而后台进程也进行了基于多进程多线程架构的大幅改进。



架构升级 – 存储虚拟化

- 存储虚拟化技术，让 K-DB彻底摆脱了第三方集群管理软件的束缚，具备了自主构建高效 RAC集群的能力。

KAS(K-DB Active Storage)



KAS功能特点

Mirroring

- 利用2-way或3-way镜像保障磁盘故障时的可用性

Striping

- 在多个磁盘中分布存储数据，实现并行磁盘I/O

Rebalancing

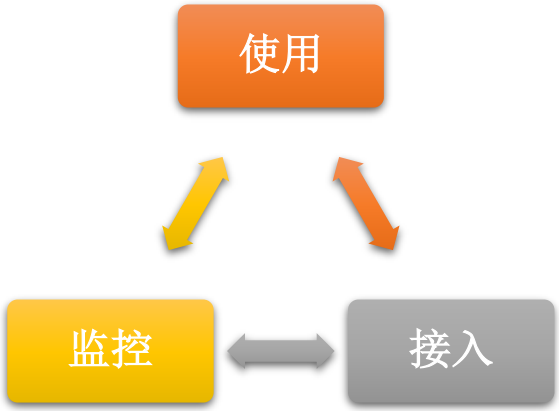
- 将数据平均存储到所有磁盘中，提高并行磁盘I/O效果

架构升级 – 三权分立/精细化审计

- 与其他数据库产品单独提供安全版数据库不同，K-DB基于安全性考虑，在最新版产品中将具体实现数据库管理的三权分立思想，丰富审计功能及体系。

三权分立

- 系统管理员
  - 系统管理员负责数据库的日常管理与使用
- 审计管理员
  - 审计管理员负责数据库的审计、内在风险分析、异常状态感知等功能
- 安全管理员
  - 安全管理员负责管理强制访问（VPD）、其他用户的数据库访问及访问权限等功能

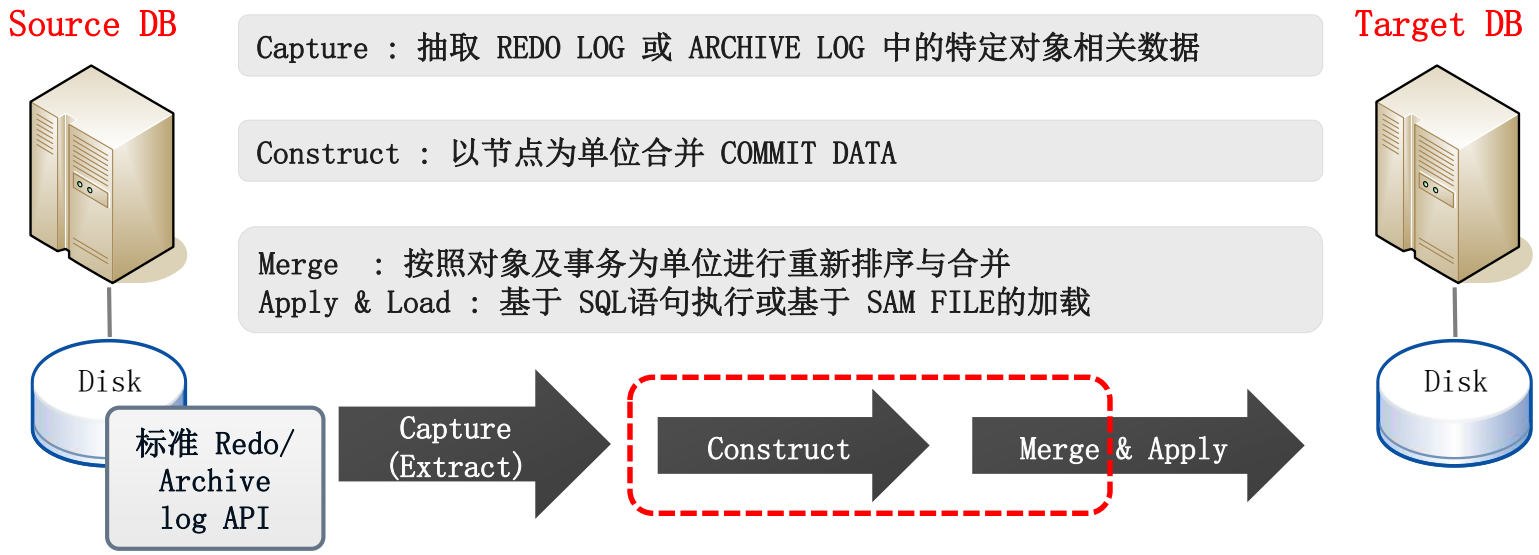


精细化审计

- 实时告警
  - 对于特定的敏感操作记录相关日志，并通过邮件或短信等方式通知客户
- 潜在问题分析
  - 对于审计日志进行自动化分类与统计，并输出分析报告
- 异常状态感知
  - 对于数据库的非法访问（如短时间内基于同一用户频繁访问敏感信息）进行自动化监测

架构升级 – CDC数据同步

- 基于 CDC工具的高性能异构数据库同步工具，更是对于客户混合数据库运营环境下数据同步问题提供了优秀的解决方案。



✓ 支持数据编辑及转换功能
→ 可以代替部分 ETL功能

支持 Source DB

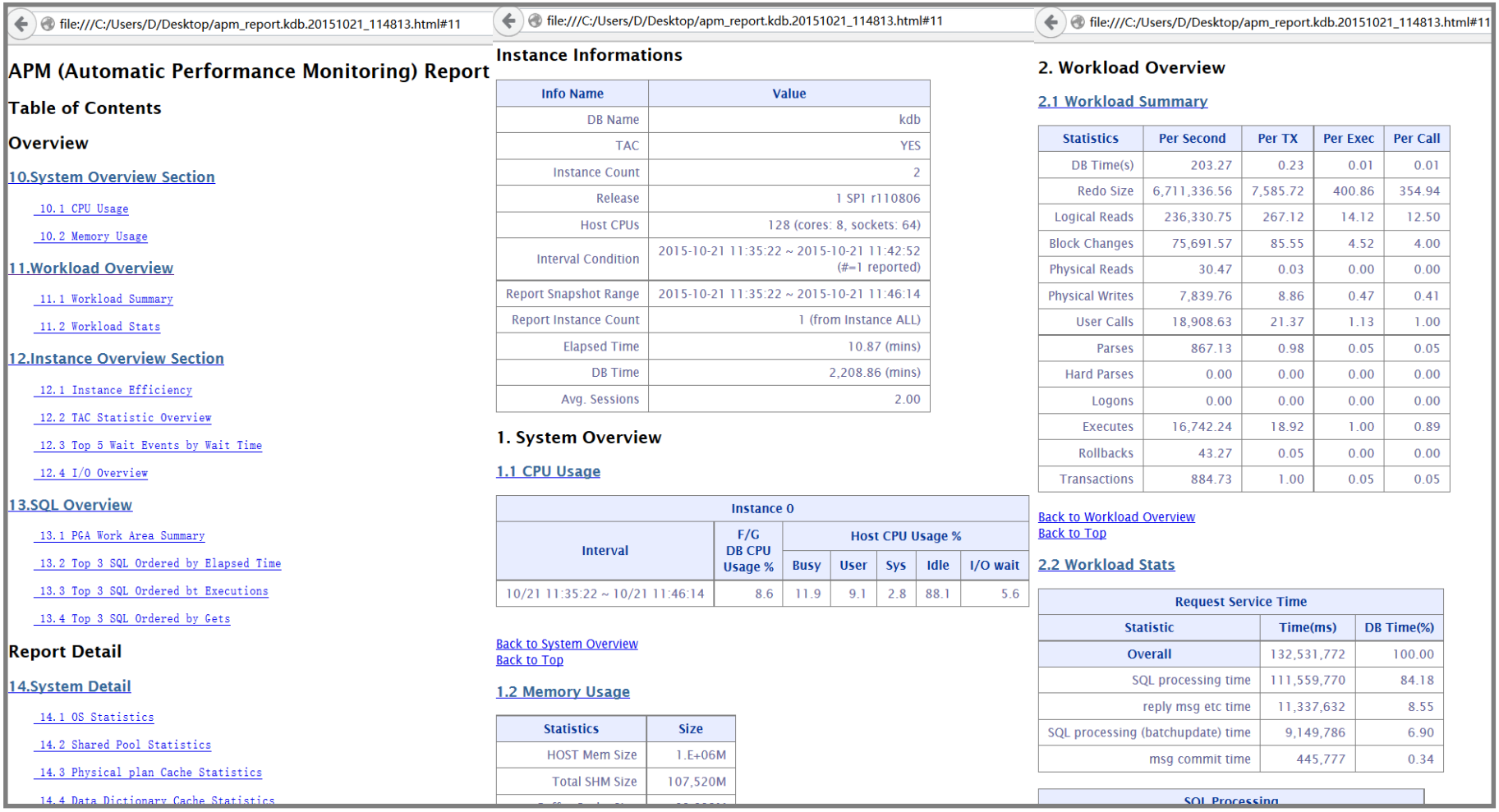
✓ K-DB (K-Sync for K-DB)
✓ Oracle (K-Sync for Oracle)

支持 Target DB

✓ K-DB
✓ Oracle
✓ Sybase
✓ DB2
✓ MS-SQL

监控方法 - APM报告(HTML)

- 基于网页版统计报告的实现，大大提升了报告的可读性，DBA不用再盯着古老的黑白屏幕了。



file:///C:/Users/D/Desktop/apm_report.kdb.20151021_114813.html#11

APM (Automatic Performance Monitoring) Report

Table of Contents

Overview

Report Detail

file:///C:/Users/D/Desktop/apm_report.kdb.20151021_114813.html#11

Instance Informations

Info Name	Value
DB Name	kdb
TAC	YES
Instance Count	2
Release	1 SP1 r110806
Host CPUs	128 (cores: 8, sockets: 64)
Interval Condition	2015-10-21 11:35:22 ~ 2015-10-21 11:42:52 (#=1 reported)
Report Snapshot Range	2015-10-21 11:35:22 ~ 2015-10-21 11:46:14
Report Instance Count	1 (from Instance ALL)
Elapsed Time	10.87 (mins)
DB Time	2,208.86 (mins)
Avg. Sessions	2.00

1. System Overview

1.1 CPU Usage

Instance 0						
Interval	F/G DB CPU Usage %	Host CPU Usage %				
		Busy	User	Sys	Idle	I/O wait
10/21 11:35:22 ~ 10/21 11:46:14	8.6	11.9	9.1	2.8	88.1	5.6

Back to System Overview
Back to Top

1.2 Memory Usage

Statistics	Size
HOST Mem Size	1.E+06M
Total SHM Size	107,520M

2. Workload Overview

2.1 Workload Summary

Statistics	Per Second	Per TX	Per Exec	Per Call
DB Time(s)	203.27	0.23	0.01	0.01
Redo Size	6,711,336.56	7,585.72	400.86	354.94
Logical Reads	236,330.75	267.12	14.12	12.50
Block Changes	75,691.57	85.55	4.52	4.00
Physical Reads	30.47	0.03	0.00	0.00
Physical Writes	7,839.76	8.86	0.47	0.41
User Calls	18,908.63	21.37	1.13	1.00
Parses	867.13	0.98	0.05	0.05
Hard Parses	0.00	0.00	0.00	0.00
Logons	0.00	0.00	0.00	0.00
Executes	16,742.24	18.92	1.00	0.89
Rollbacks	43.27	0.05	0.00	0.00
Transactions	884.73	1.00	0.05	0.05

Back to Workload Overview
Back to Top

2.2 Workload Stats

Request Service Time		
Statistic	Time(ms)	DB Time(%)
Overall	132,531,772	100.00
SQL processing time	111,559,770	84.18
reply msg etc time	11,337,632	8.55
SQL processing (batchupdate) time	9,149,786	6.90
msg commit time	445,777	0.34

SQL Processing

监控方法 – 图形化实时监控

inspur 浪潮

K-APM 数据库系统

实时监控平台

备份/恢复 - RMGR & Flashback

- RMGR终于不只是架子，不仅能够实现数据库的灵活备份与恢复，进一步提供了数据块改动检测功能。而闪回功能，已经实现了数据库级别的恢复。

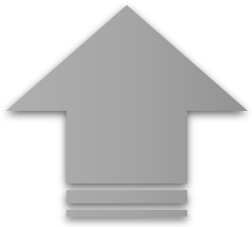
RMGR备份恢复工具

- RMGR 基本功能
  - Online Full Backup
  - Incremental Backup
  - Block Change Tracking
  - Automatic Recovery
  - Tablespace/Datafile级别备份及恢复
  - 恢复 Auxiliary Database

```
$ kdrmgr backup -i -o /home/winds/backup2 -b /home/winds/backup/rmgr.inf -v
============================================================
= Recovery Manager(RMGR) starts                            =
=                                                          =
= Inspur Corporation Copyright (c) 2008-. All rights reserved. =
============================================================
============================================
  RMGR - INCREMENTAL backup
============================================
DB connected
DBNAME: kdb
```

闪回功能升级

Flashback Database



Flashback Query

运维方案

统计分析

TOP Session
TOP SQL
TOP Wait
TOP Program
TOP Module
TOP User
Lock Tree

自动化分析

自动化监控

系统优化

SQL优化
表结构优化
索引优化
参数优化
修改备份策略
集群架构优化

自动化报表

发现问题

业务性能低下
触发阈值

实时监控

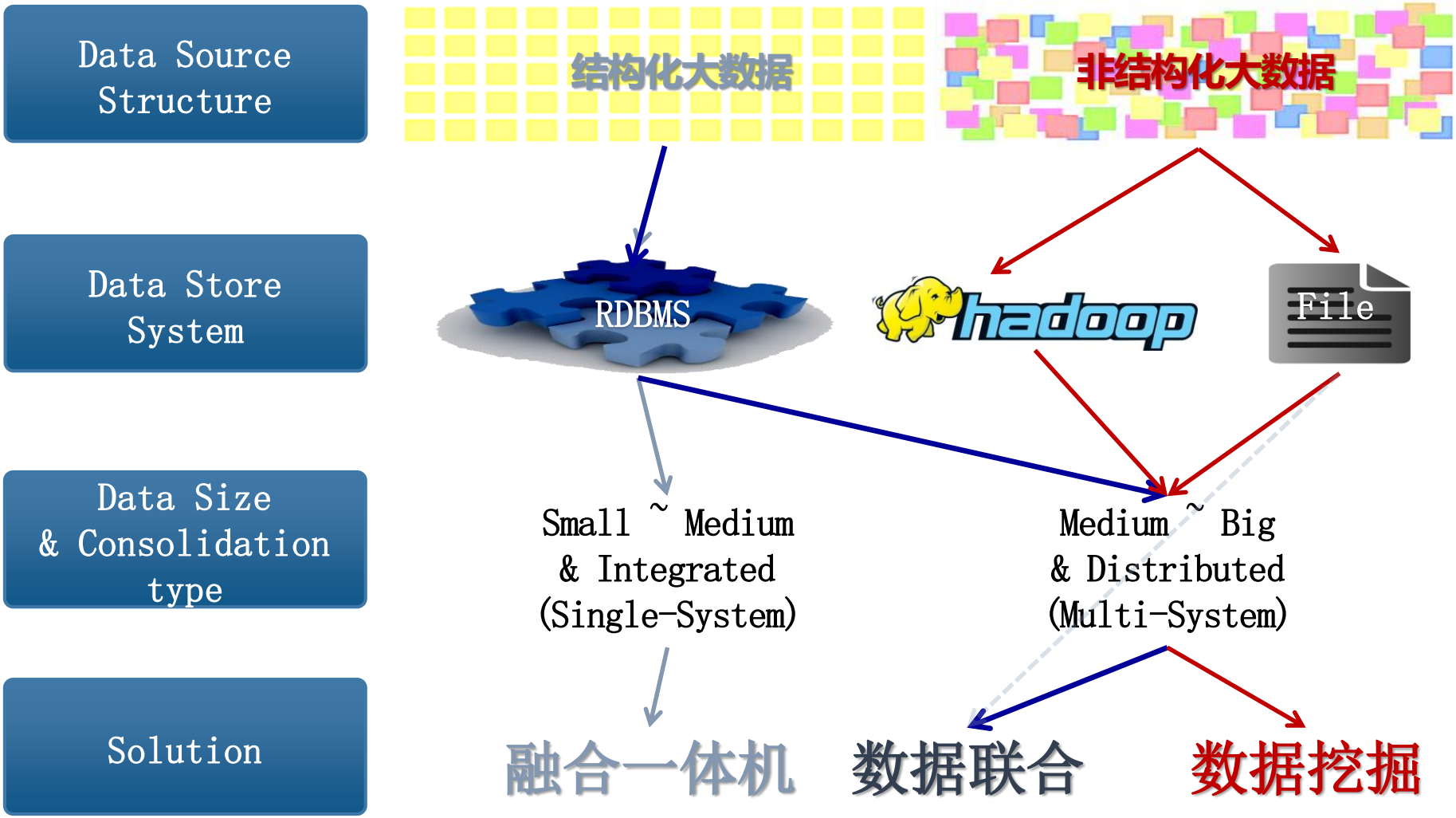
集群状态监控
会话状态监控
告警信息监控
锁信息监控
空间信息监控
Disk I/O监控
CPU、内存监控

巡检报告

阶段统计报告
自定义报告

新的挑战

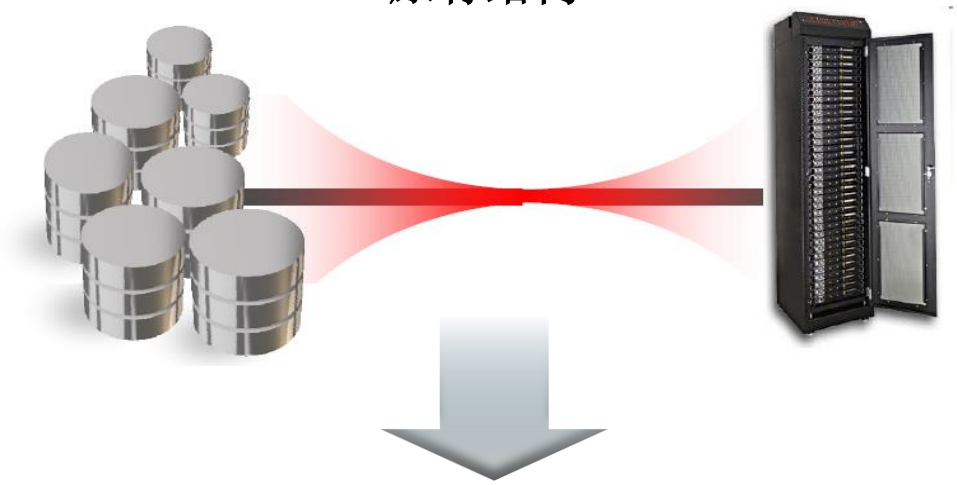
大数据环境下的新挑战


Data Source Structure
Data Store System
Data Size & Consolidation type
Solution
结构化大数据
非结构化大数据
RDBMS
hadoop
File
Small ~ Medium & Integrated (Single-System)
Medium ~ Big & Distributed (Multi-System)
融合一体机
数据联合
数据挖掘

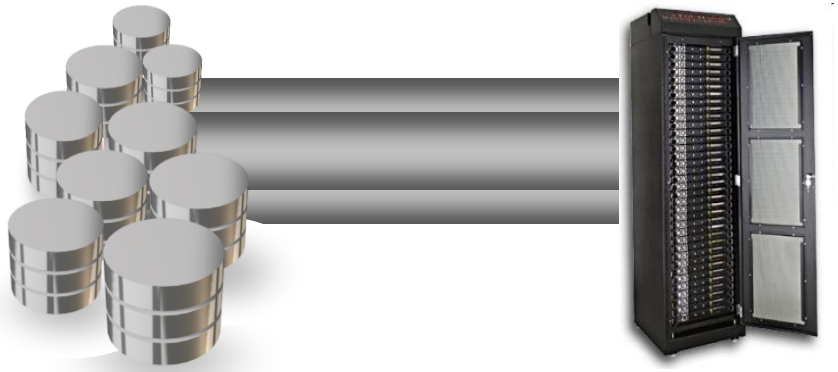
融合一体机

传统数据库瓶颈与一体机解决方案

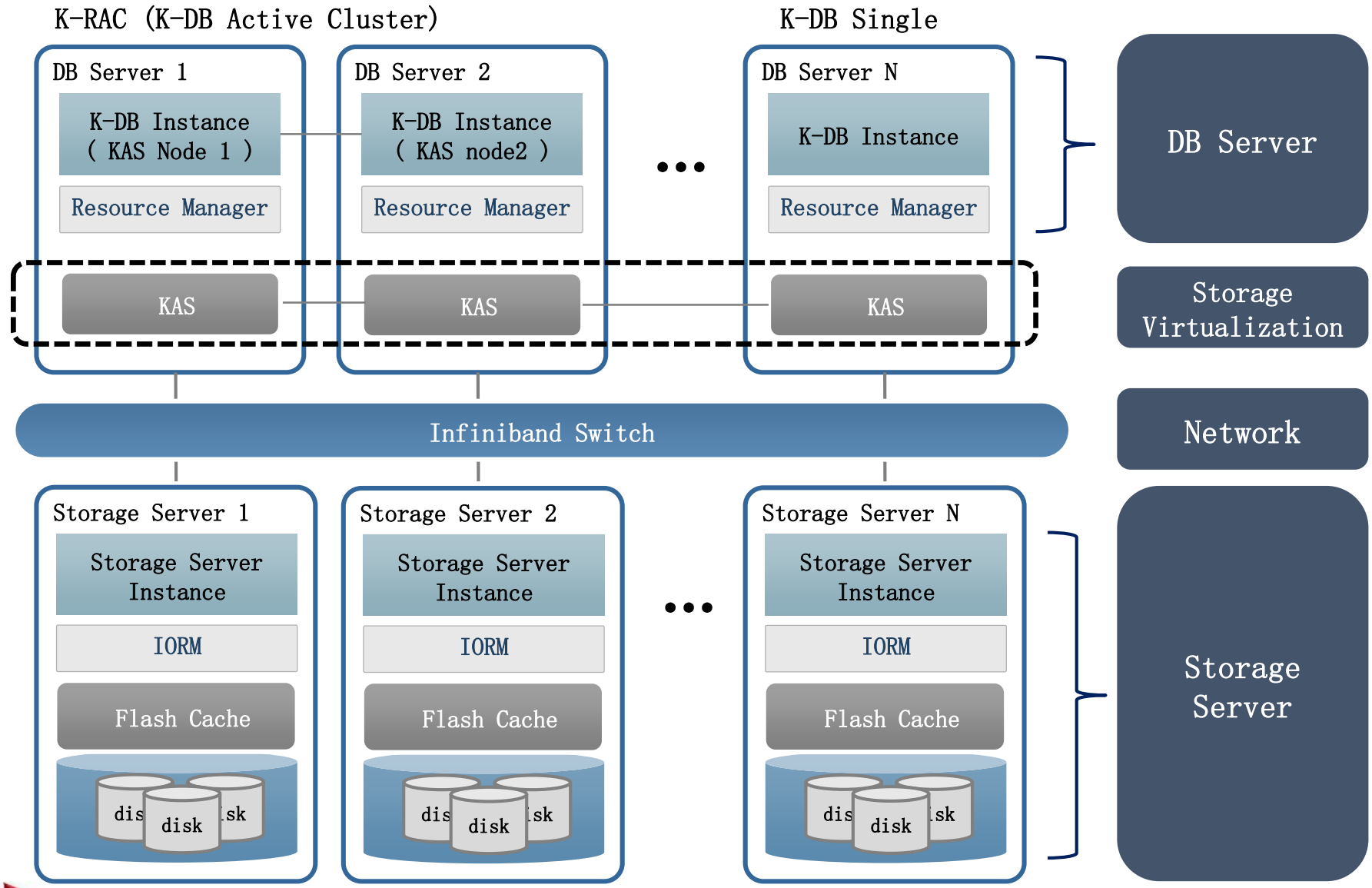
原有结构



一体机解决方案

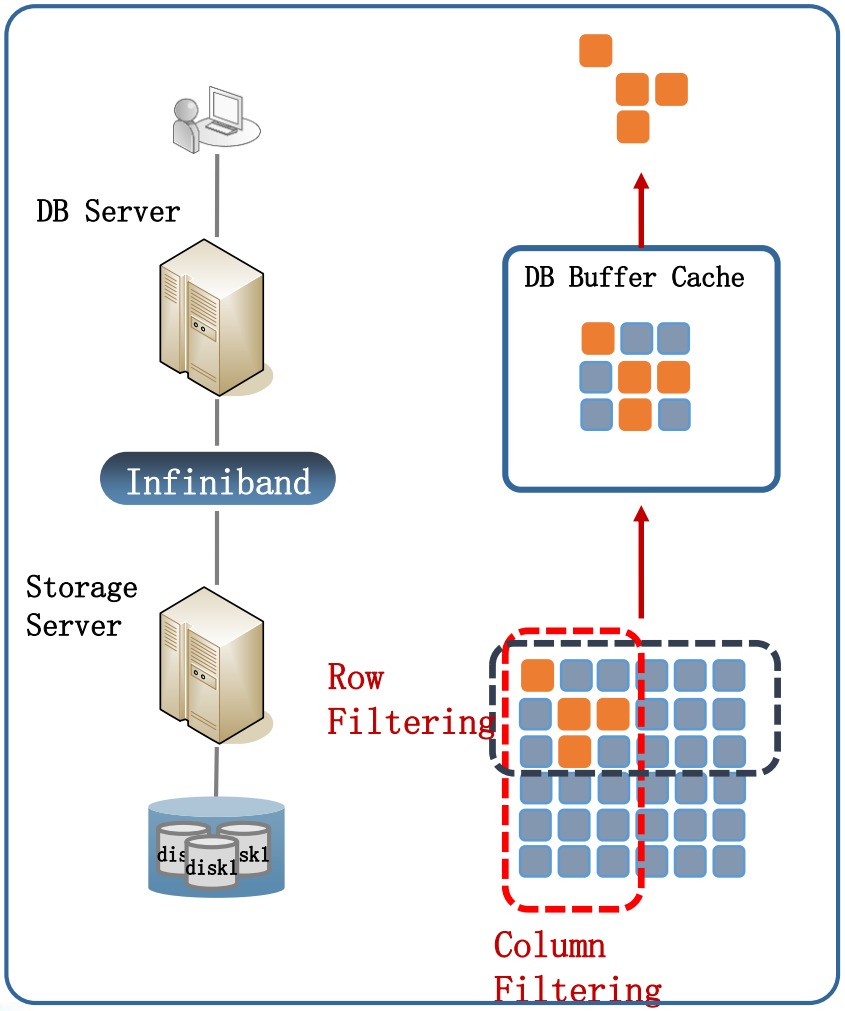


New Architecture



Smart Filtering

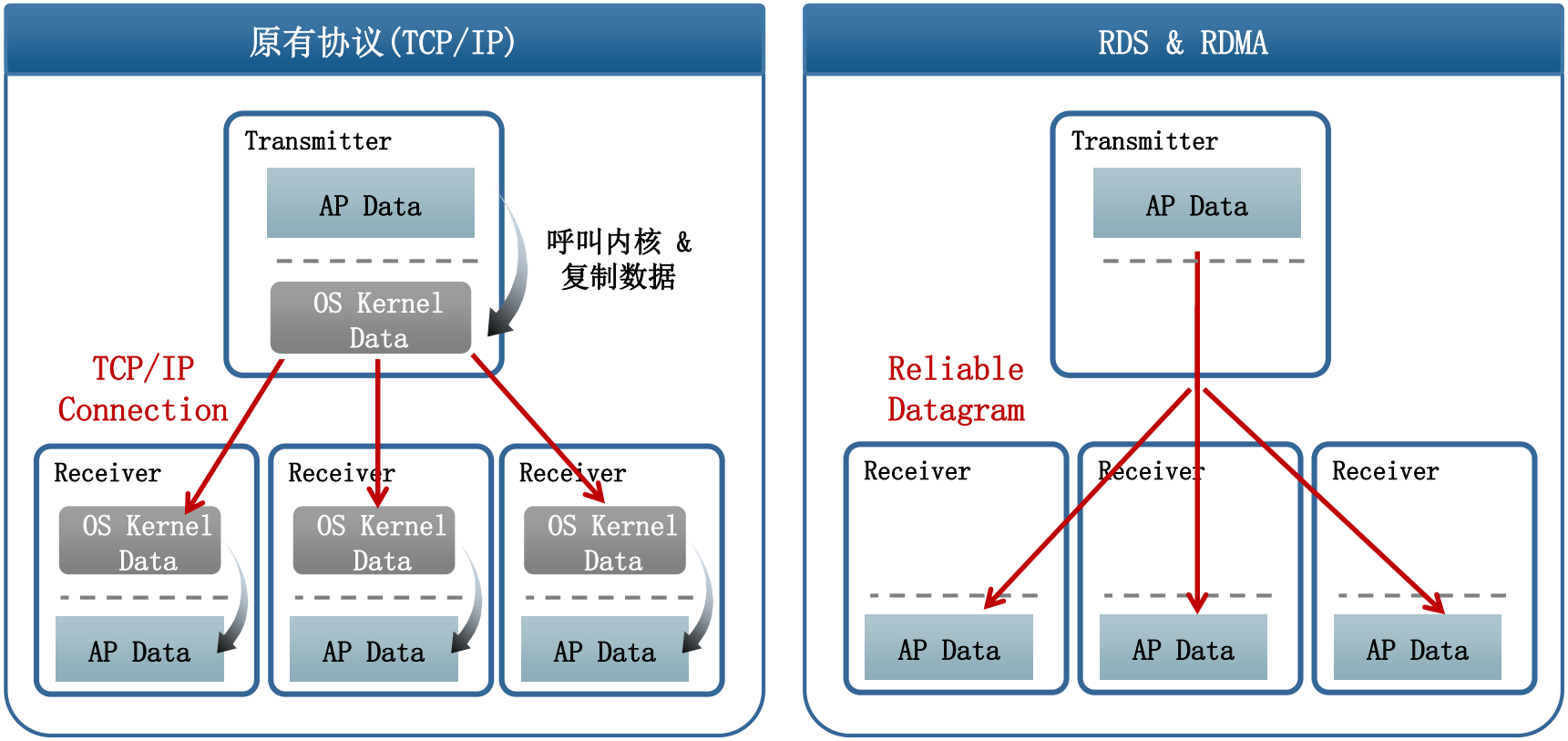
- 从存储服务器中只提取必要的数据传送到DB，减少DB的作业负担，缩短数据处理时间



技术	内容
Smart Scan	在存储设备中只抽取符合SQL条件Row, 和Column 传输到DB → 减少DB的数据处理负担
Storage Index	把储存在磁盘的数据概要信息实时进行管理 → 忽略不符合SQL条件的Disk I/O

RDS & RDMA

- 使用为了大量数据的高速传输的 RDS&RDMA 协议→ Low Overhead低开销

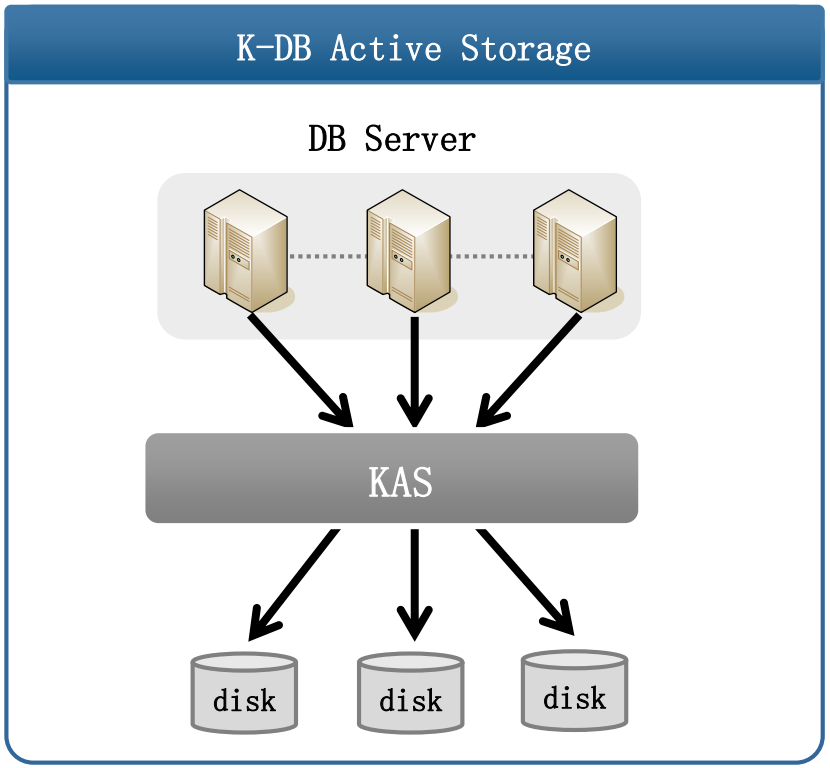


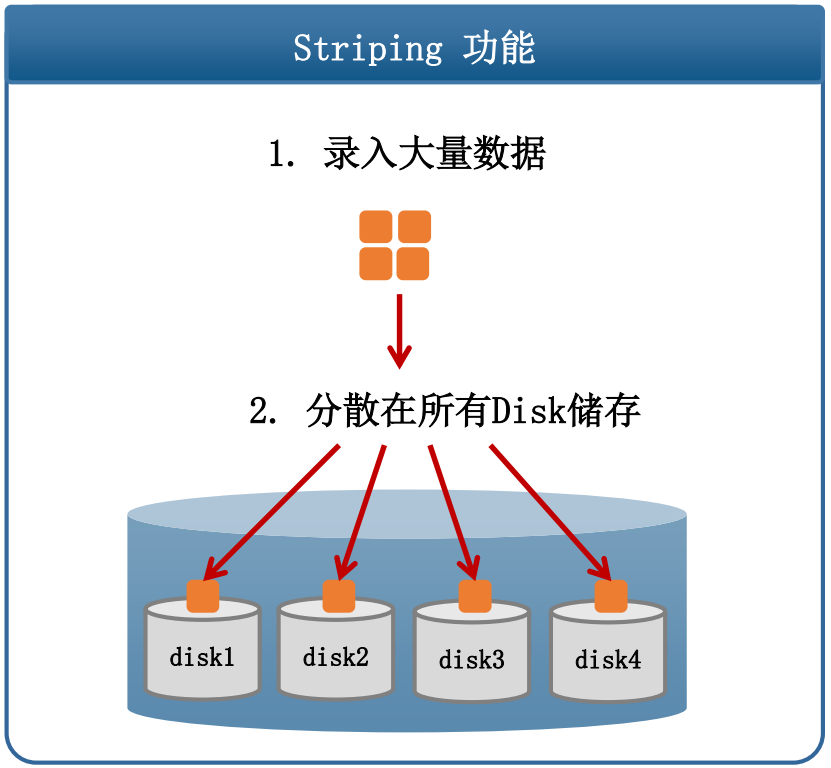
- RDS (Reliable Datagram Socket) : 可信的 1:N传输协议，并且对比TCP/IP 减少负荷
- RDMA (Remote Direct Memory Access)

  :直接参照 AP领域的内存调用，避免OS间接访问内核 →减少CPU负荷并且可以简短Latency（延时）

KAS(K-DB Active Storage)

- 管理分散储存在各个磁盘上的数据信息,通过 Disk I/O并行处理提高性能

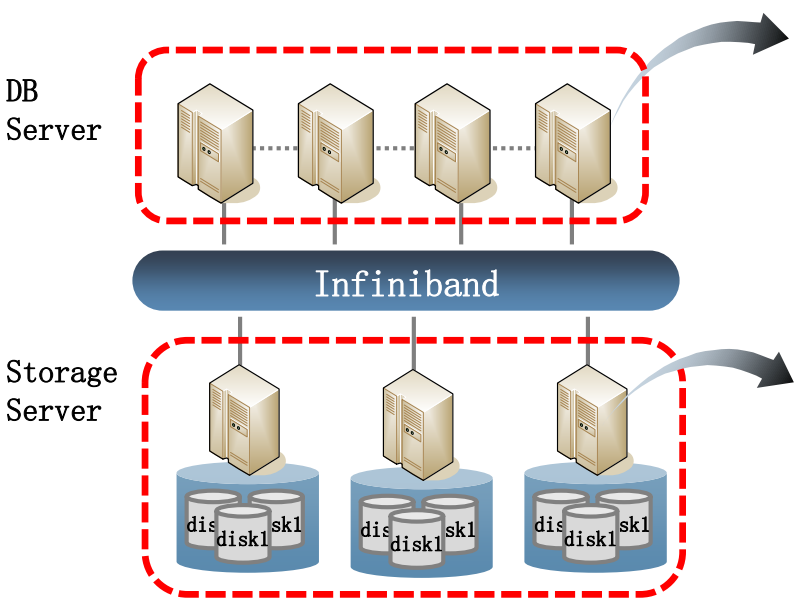




- KAS :管理分散储存在各个磁盘上的Clustered File System
  → 并行Disk I/O : 把数据分散储存在各个Disk提高I/O 性能

Resource Management

- 根据业务分配CPU和Disk I/O 的资源，业务负荷大的状况下，可以保障性能和有效使用资源



Resource Manager

- 分配 DB 服务器的 CPU资源到各个业务
- 根据业务、会话 分配CPU，从而保障 DB 性能

IORM

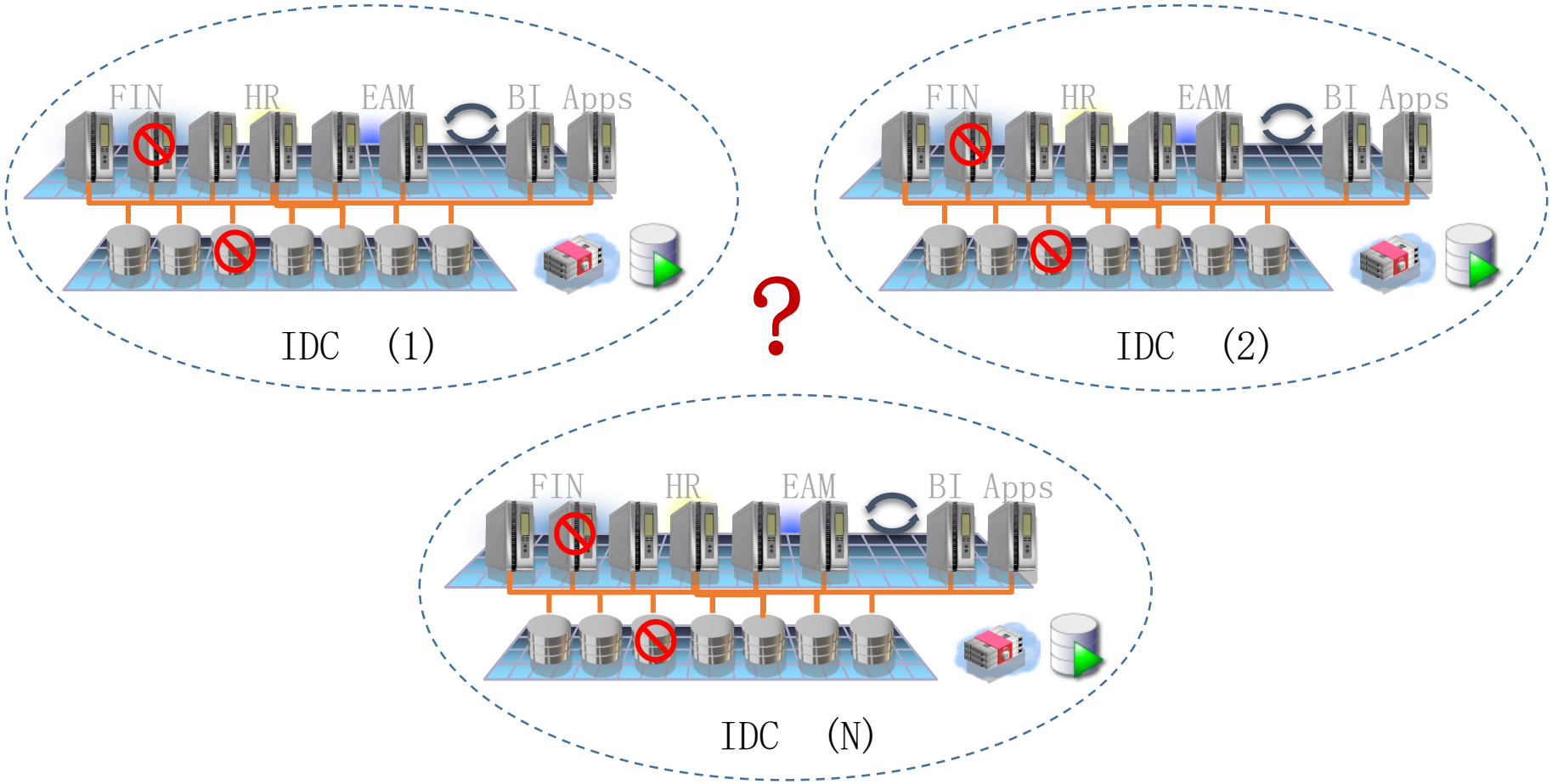
- 把存储服务器的Disk I/O分配到业务
- 在高负荷状况下防止数据过度等待

- 把DB服务器的CPU和存储服务器的Disk I/O分配到各个业务上保障性能
- 重要的业务在DB和存储服务器高负荷时也能充分保障性能

数据联合

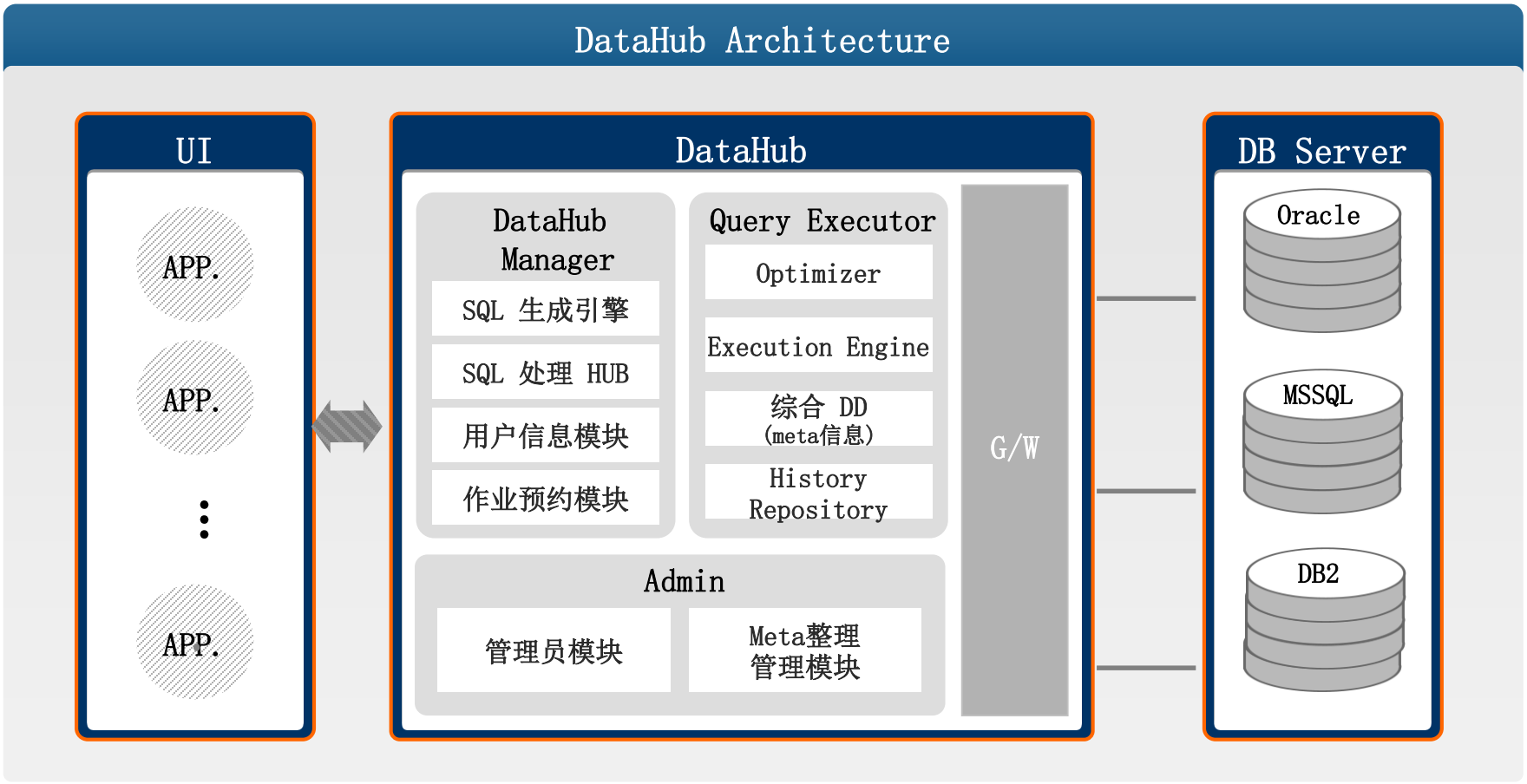
分布式数据中心的数据联合

- 在地域分布、上百上千的数据中心需要联合分析计算的场景下，需要寻求新的数据架构



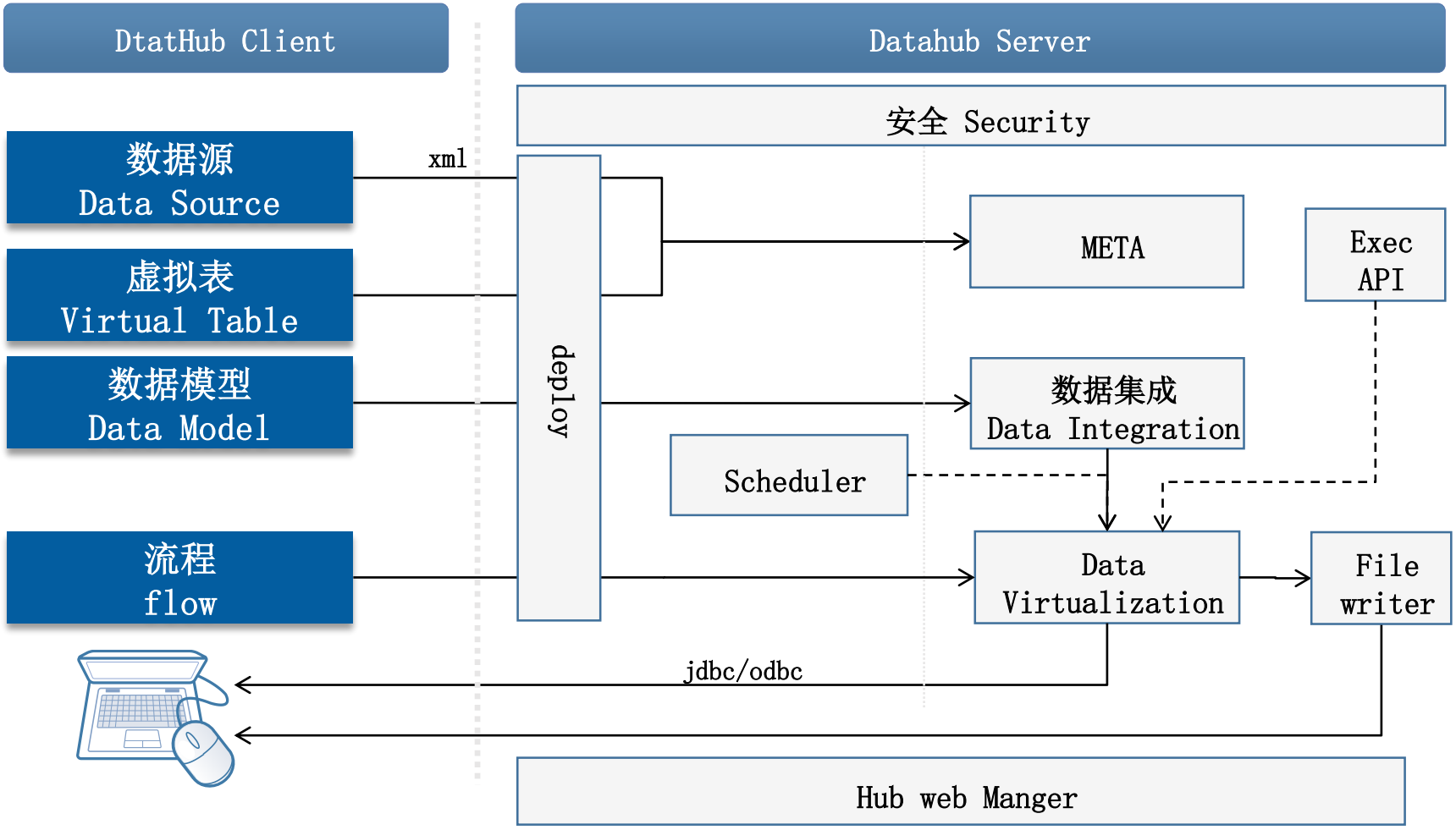
Data Hub架构

- Data Hub可以通过单纯连接已有的多个异构数据库系统，便能够轻松构建跨数据库、跨地域的集群数据分析系统。



Data Hub架构

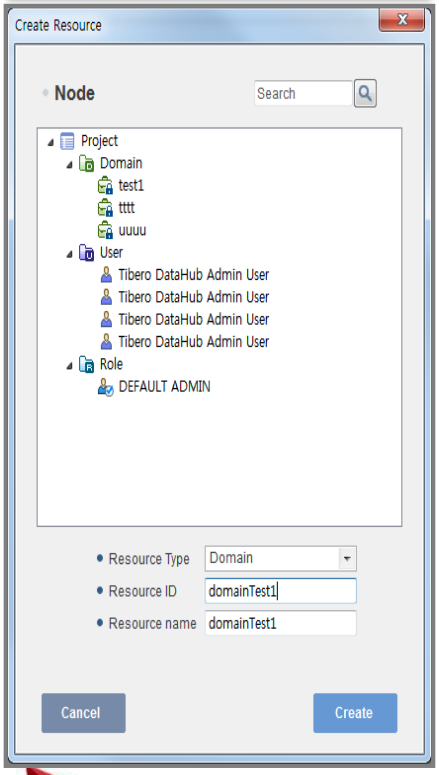
- 通过数据源镜像、数据对象建模、开发业务流这种层层递进的开发模式，Data Hub可以实现对于后端海量数据库集群的模拟与数据抽取。

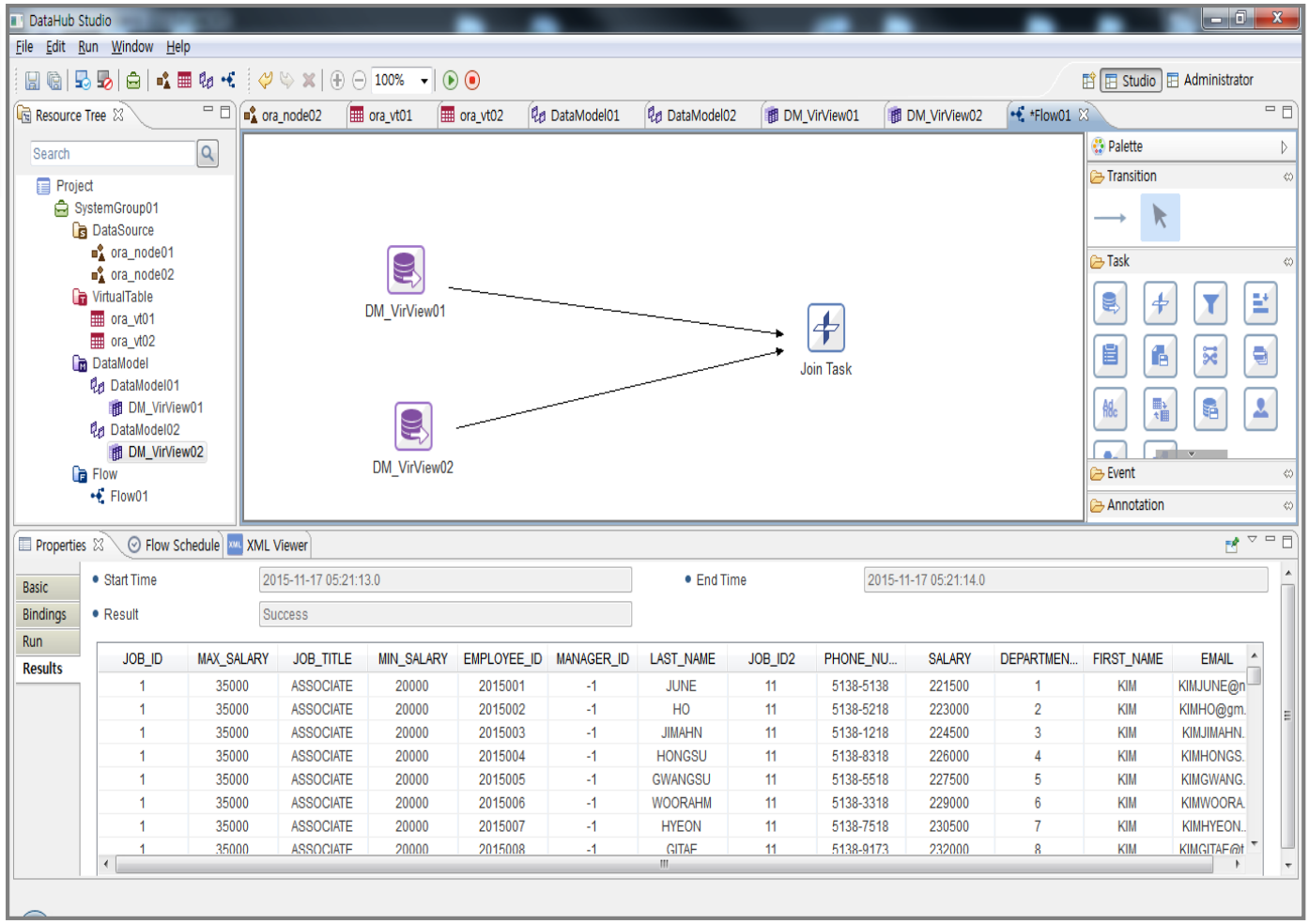


Data Hub 业务开发工具

- 基于 GUI 拖拽方式的业务开发模式，使后期数据管理及分析业务的开发无比轻松。

Data Hub Studio





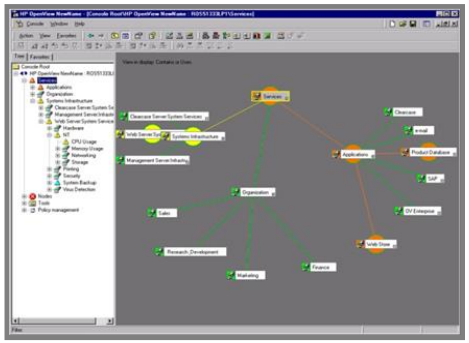
全新架构下的运维

全新架构特点

软件 + 硬件 + 跨平台

全新运维需求


+

+



谢 谢 !