KunlunBase

KunlunBase 分布式任务协调框架介绍

summer

泽拓科技（深圳）有限责任公司

KunlunBase

目 录

COMPANY

01 需求及背景

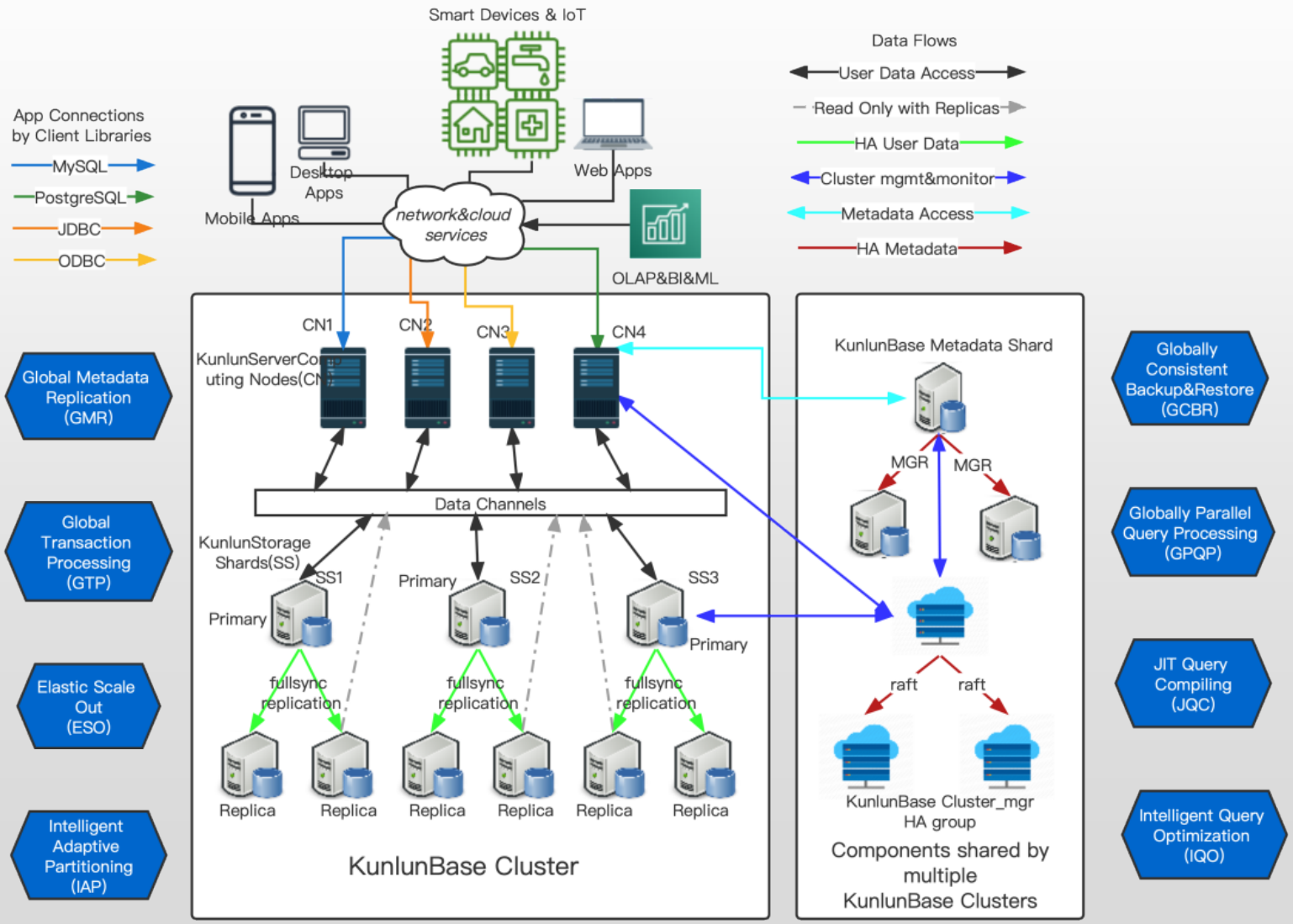
02 架构介绍

03 复杂任务编排

04 可观测性构建

http://www.kunlunbase.com

01 需求及背景



http://www.kunlunbase.com

需求及背景

分布式数据库任务管理的核心诉求

02 架构介绍--抽象任务模型

- 多个集群任务的并发执行
- 单个任务由多个不同的操作序列组成
- 不同的操作序列间，有不同的时序依赖
- 多个集群任务由不同的时序依赖
- 单节点任务执行并发度的控制

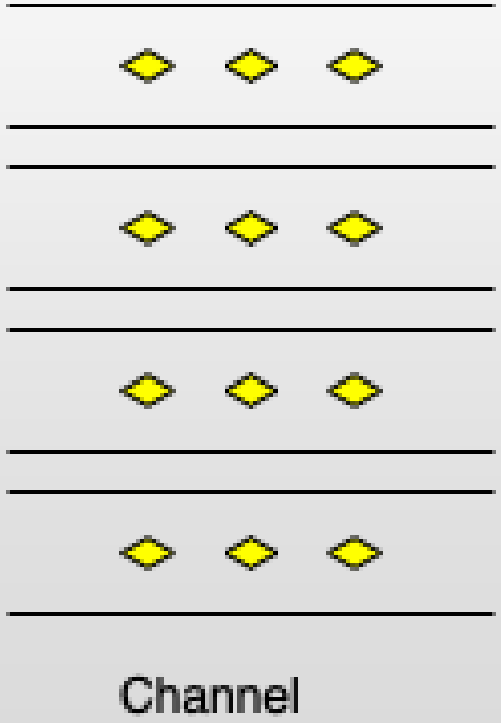
宏观目标：

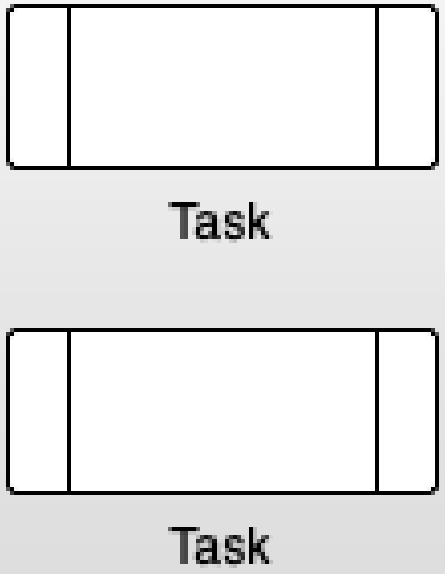
- ✓ **用统一的框架来描述集群各物理节点间有时序依赖的任务的执行**
- ✓ **尽可能的降低编程难度**
- ✓ **随着功能及需求的变化易于扩展**

KunlunBase
http://www.kunlunbase.com

架构介绍--抽象任务模型

抽象业务模型



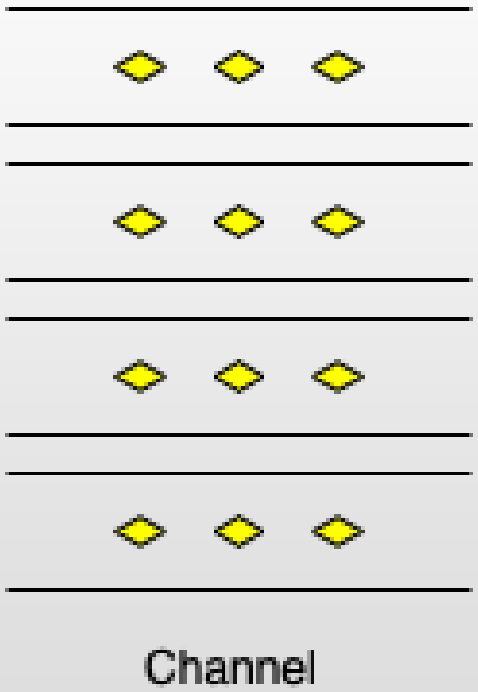


http://www.kunlunbase.com

02 架构介绍--channel

Channle -- 一次远程 RPC 调用

- 阻塞方式等待应答
- 应答包含 RPC 调用的结果信息，成功或者失败
- 超时即报错返回。不包含重试逻辑，由上层控制



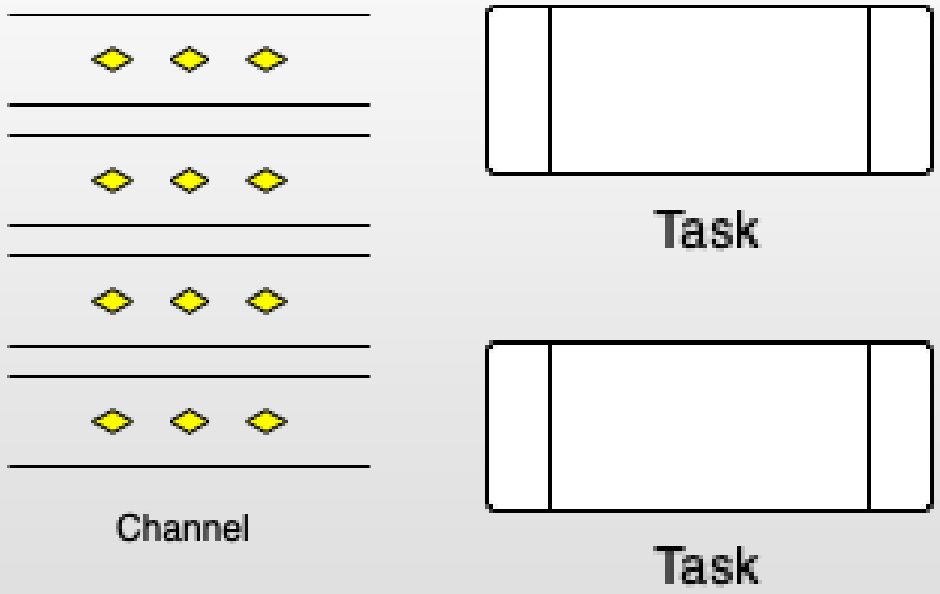
- 对指定的数据库实例执行一次数据备份。
- 对某个数据库实例执行一次变量设置。
- 拉取一次物理节点的信息。

http://www.kunlunbase.com

架构介绍--Task

Task -- 多次远程 RPC 的组合

- 包含一个或者多个 Channel 实例
- Task 负责对 Channel 发起 RPC 调用
- Task 采用半同步方式管理 Channel 组
- Channel 间并发执行
- 所有 Channel 应答成功，Task 成功

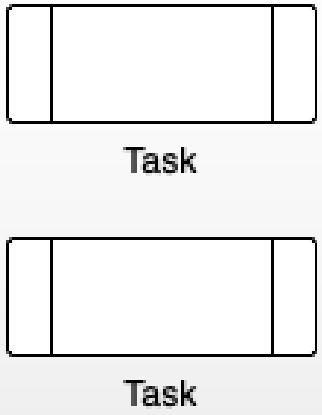


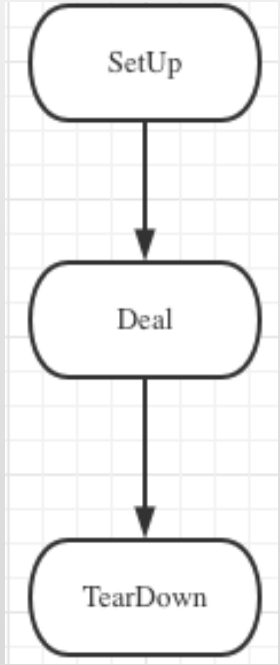
➢ 创建 Shard ,包含多个 channel, 每个 channel 独立负责一个存储节点的创建。

架构介绍--Mission

Mission -- 多个 Task 的组合，表示一个独立的功能

- 包含一个或者多个Task实例
- 多个 Task 间执行采用 串行方式
- 多个 Mission 间的执行，串行预处理/并发执行



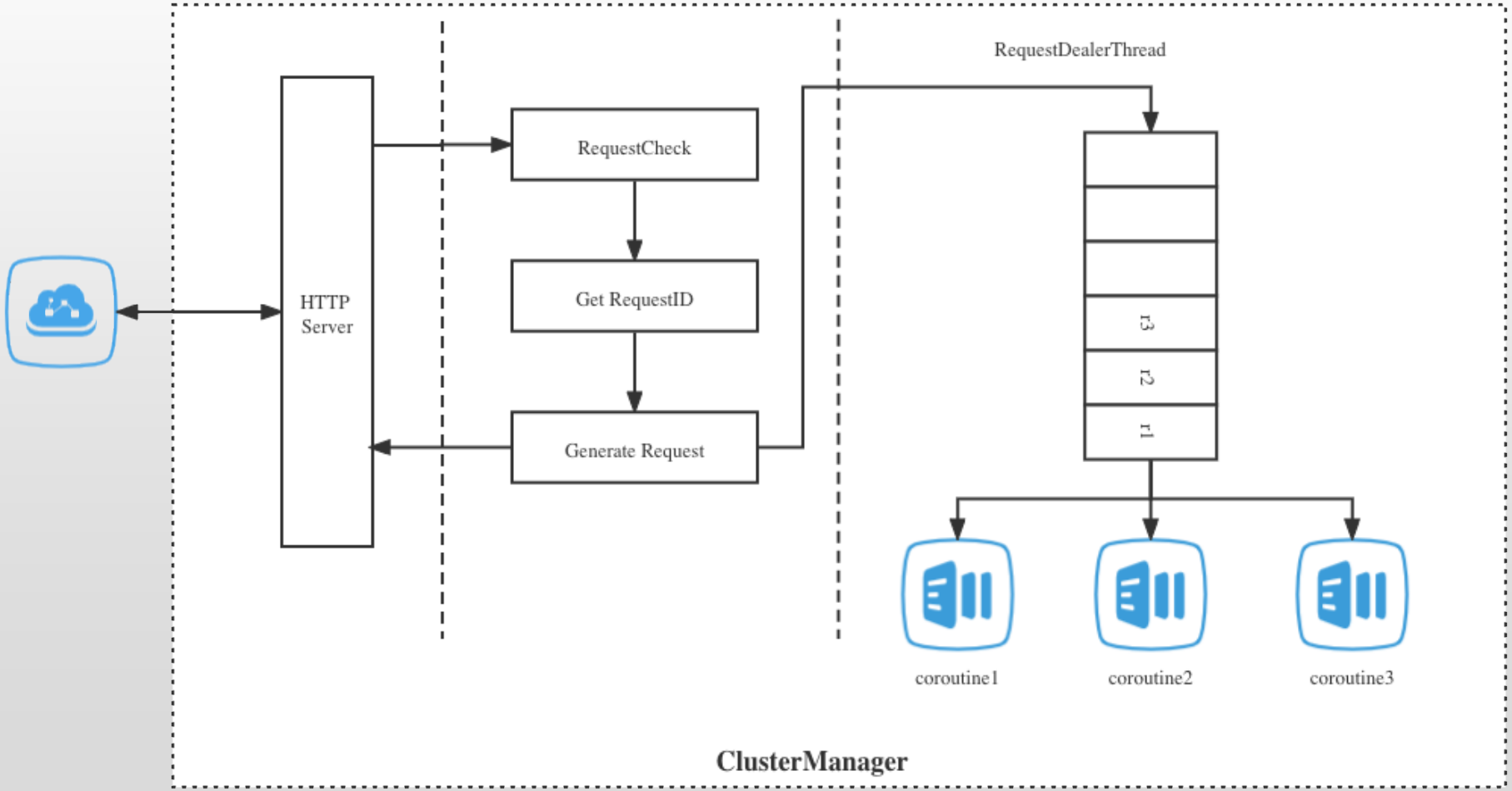


➢ 扩容任务，
1. 全量数据导出Task
2. 全量数据导入 Task
3. 增量日志回放 Task
4. 元数据路由信息修改 Task

02

架构介绍--整体架构

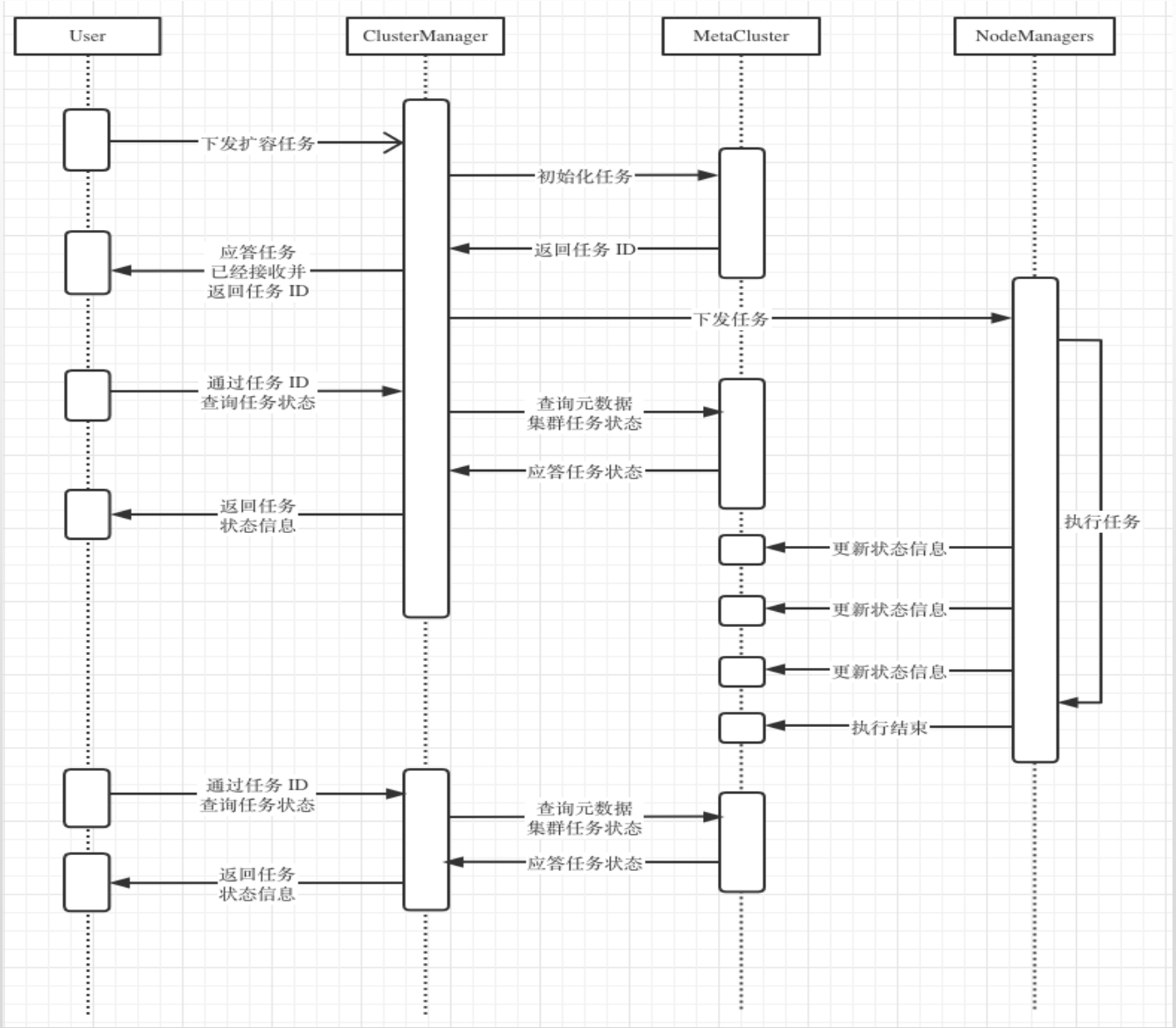
架构模型



http://www.kunlunbase.com

架构介绍--交互模型

交互模型



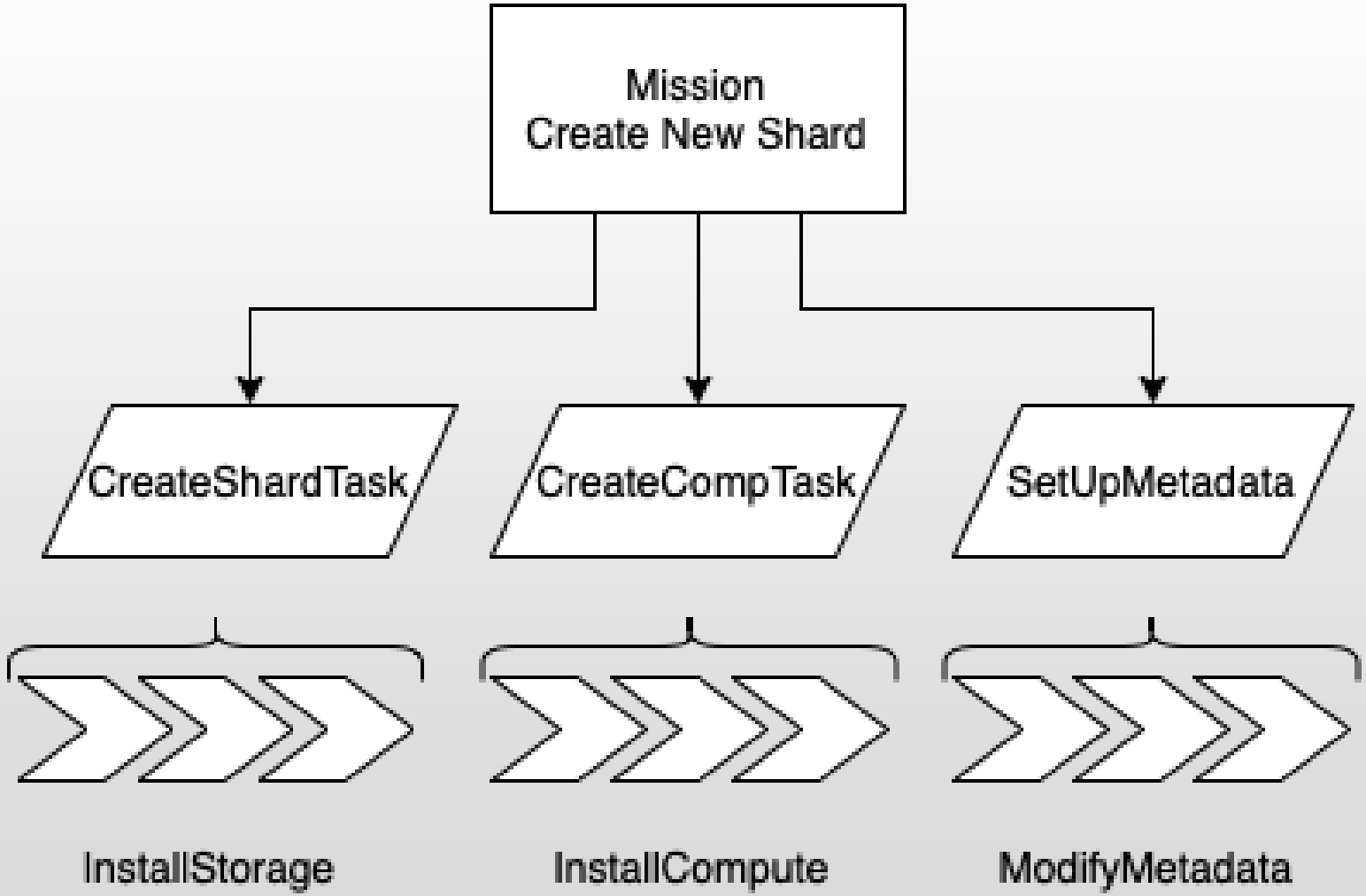
http://www.kunlunbase.com

03

复杂任务编排--集群按时间点回档

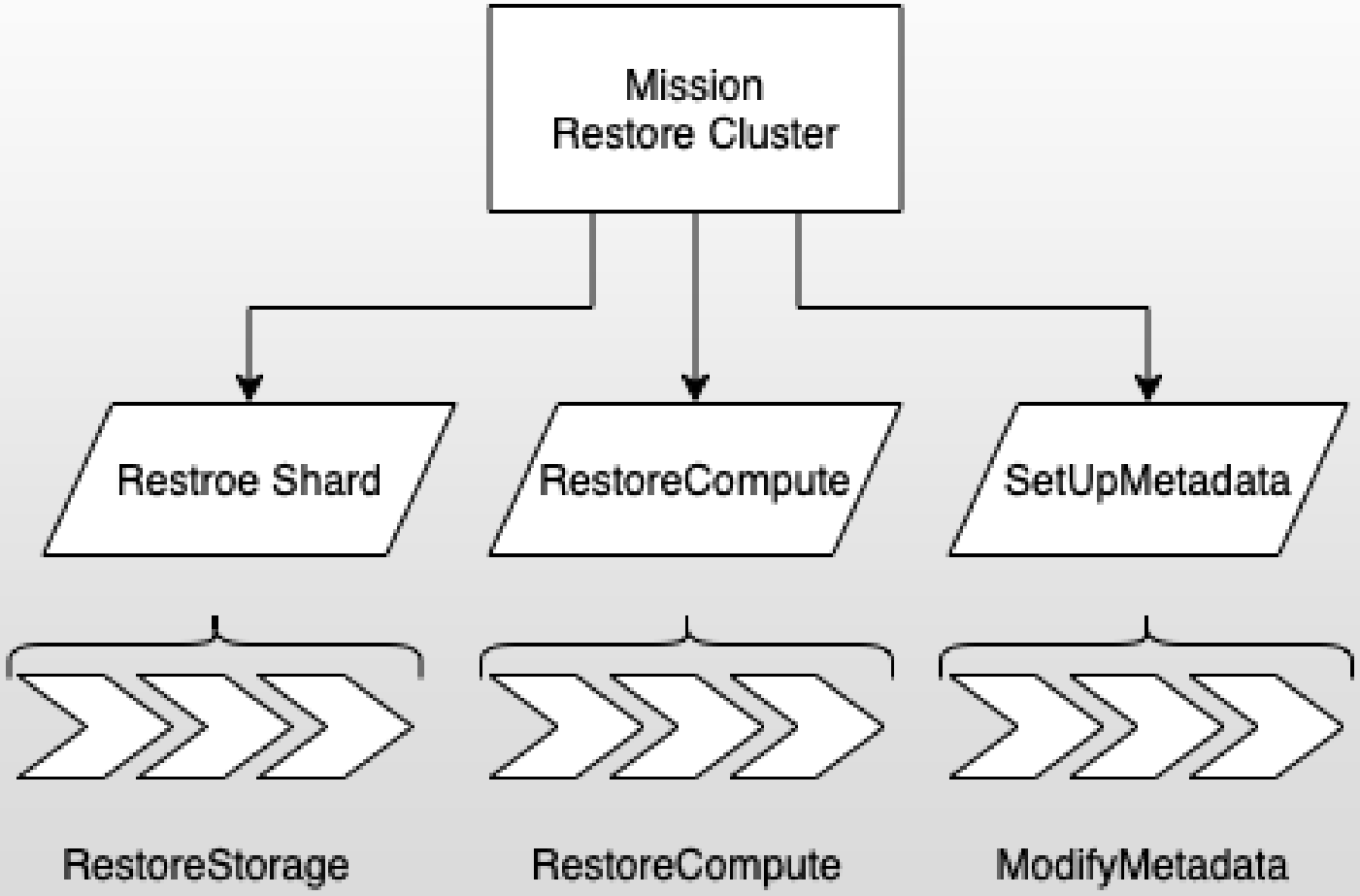
1. 创建同规格的新 cluster
2. 对新集群的所有 shard 调用回档工具
3. 对新集群的所有计算节点调用回档工具
4. 更新新集群的所有的元数据信息

03 复杂任务编排--createCluster



KunlunBase
http://www.kunlunbase.com

03 复杂任务编排--restoreCluster

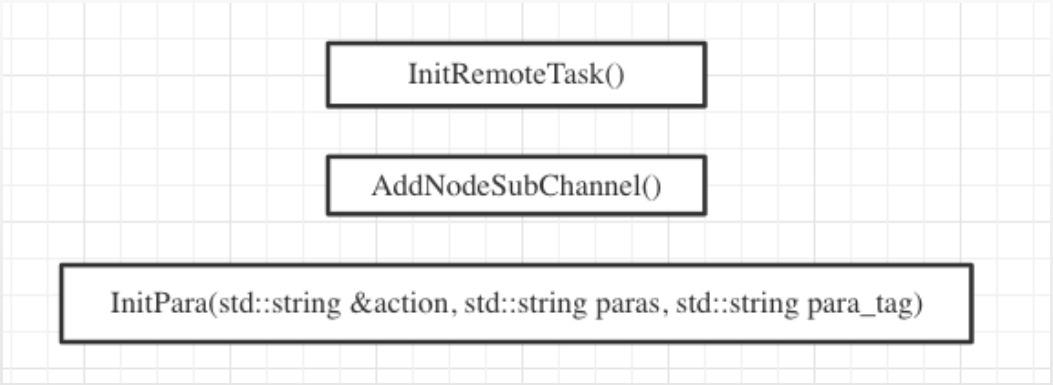


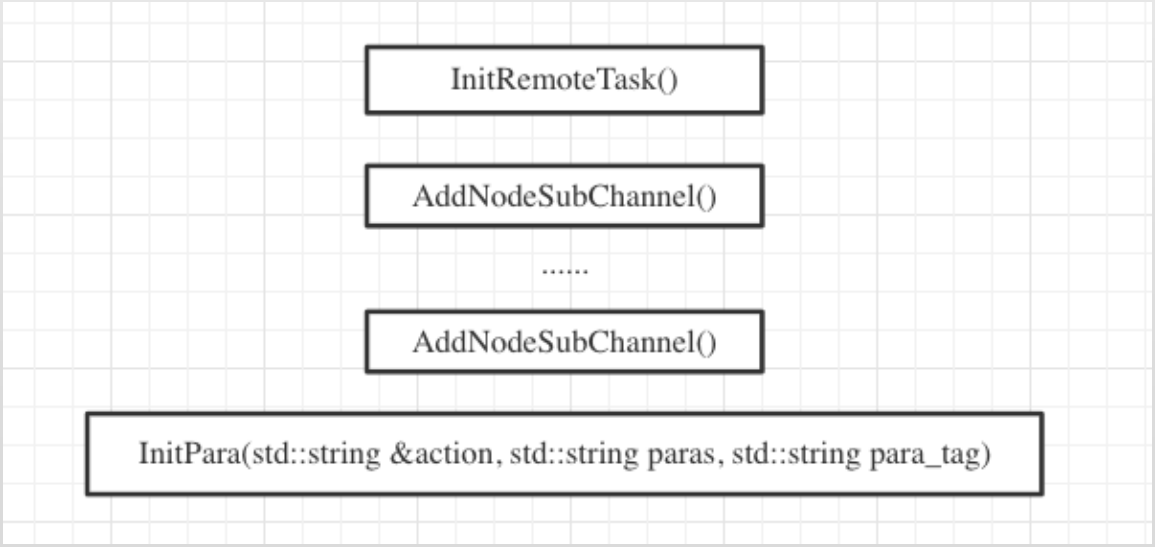
http://www.kunlunbase.com

复杂任务编排--编程实践

KunlunBase

Task 编排



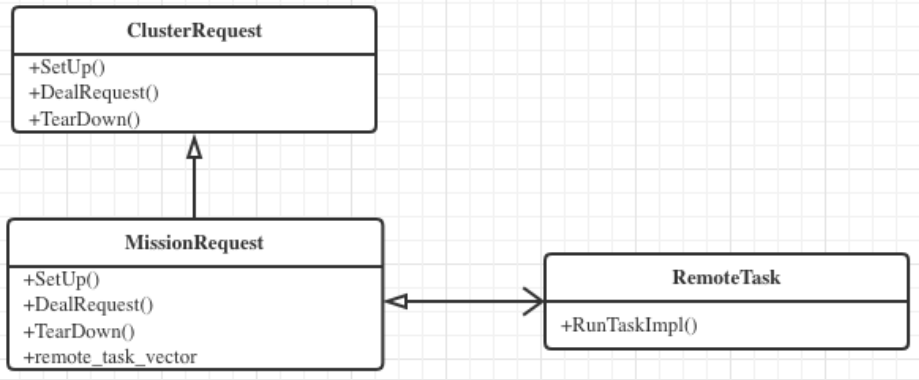



```
class RemoteTask : public kunlun::ErrorCup {

public:
  RemoteTask();
  ~RemoteTask();

private:
  RemoteTask(const RemoteTask &) = delete;
  RemoteTask &operator=(const RemoteTask &) = delete;

public:
  bool InitRemoteTask(int timeout_sec) ;
  bool AddNodeSubChannel(brpc::Channel *sub_channel);
...

  // sync run in bthread
  bool RunTaskImpl() ;
  void RecordeSubChannleReturnValue(const brpc::Controller *cntl, int index) ;
  bool AllSubChannelSuccess() ;
  void InitPara(std::string &action, std::string &paras, std::string &para_tag);
private:
  brpc::ParallelChannel remote_channel_;
...
  std::string action_;
  std::string action_paras_;
  std::vector<std::string> sub_channel_return_info_vec_;
  rapidjson::Document return_info_json_array_;

  // for multi nodemanager action has different paras,
  // use this tag to get the right paras
  std::string action_paras_tag_;
};
```

复杂任务编排--编程实践

KunlunBase

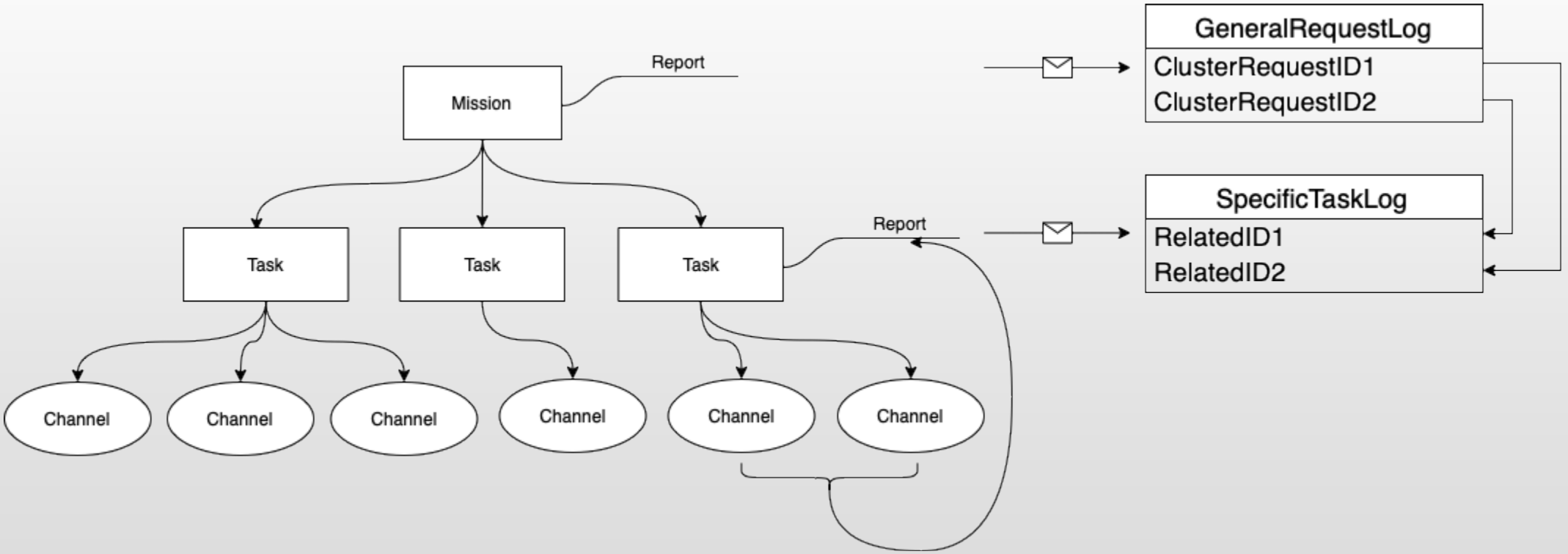
Mission 编排




```
class MissionRequest : public ClusterRequest {
  typedef ClusterRequest super;
public:
  MissionRequest(google::protobuf::RpcController *cntl_base,
                 const HttpRequest *request, HttpResponse *response,
                 google::protobuf::Closure *done)
      : super(cntl_base, request, response, done) {
    task_manager_ = nullptr;
  }
  virtual ~MissionRequest();
  virtual void SetUpImpl() override final;
  // user should add arrange remote task logic
  virtual bool ArrangeRemoteTask() = 0;
  // user shold add setup logic here
  virtual bool SetUpMisson() = 0;
  virtual void DealRequest() override final;
  virtual void TearDownImpl() override final;

private:
  TaskManager *task_manager_;
};

void MissionRequest::SetUpImpl() {
  ArrangeRemoteTask();
  SetUpMisson();
}

void MissionRequest::DealRequest() {
  // do the request iterator vec
  auto &task_vec = task_manager_->get_remote_task_vec();
  auto iter = task_vec.begin();
  for(;iter != task_vec.end();iter++){
    bool ret = (*iter)->RunTaskImpl();
    if (!ret){
      setErr("%s",(*iter)->getErr());
      return false;
    }
  }
  return true;
}

void MissionRequest::TearDownImpl() {}
```

http://www.kunlunbase.com

04 可观测性构建



http://www.kunlunbase.com

KunlunBase

Thank you

Q & A