超大规模数据挖掘架构及方法论

更深入地洞察
更科学地决策

议题

- 超大规模数据挖掘架构
  - 主流并行计算架构
  - 数据挖掘并行特点
  - GDM并行架构
- 数据分析方法论

议题

- 超大规模数据挖掘架构
  - 主流并行计算架构
  - 数据挖掘并行特点
  - GDM并行架构
- 数据分析方法论

MPI

设计目标	(1)设计一套应用程序接口，使编程者不需要面对编译器、操作系统执行包。 (2)允许高效通信：避免内存到内存的拷贝（memory-to-memory copying），允许overlap计算和通信，任务分解到各线程。 (3)支持异构环境。 (4)采取可靠的通信接口：编程者不用应对通信失败。 (5)支持多用户平台，而不需要明显的程序修改。 (6)线程安全。

```
//java版MPI框架MPJ示例代码
public static void main(String[] args) {
    // 测试并行
    long t1 = System.currentTimeMillis();
    int n=20;
    double sum=0;
    MPI.Init(args);    //初始化
    int me = MPI.COMM_WORLD.Rank(); //线程编号
    int size = MPI.COMM_WORLD.Size(); //线程数
    for(int i=me;i<n;i+=size){
        sum+=i;
    }
    MPI.Finalize();    //并行结束
    System.out.println("process"+me+":t="+(System.currentTimeMillis()-t1)+":sum="+sum);
}
```

缺点：

串行代码cpu、内存资源重复占用；

➢ MPI基本函数

MPI_INIT:	启动MPI环境
MPI_COMM_SIZE:	确定进程数
MPI_COMM_RANK:	确定自己的进程标识符
MPI_SEND:	发送一条消息
MPI_RECV:	接收一条消息
MPI_FINALIZE:	结束MPI环境

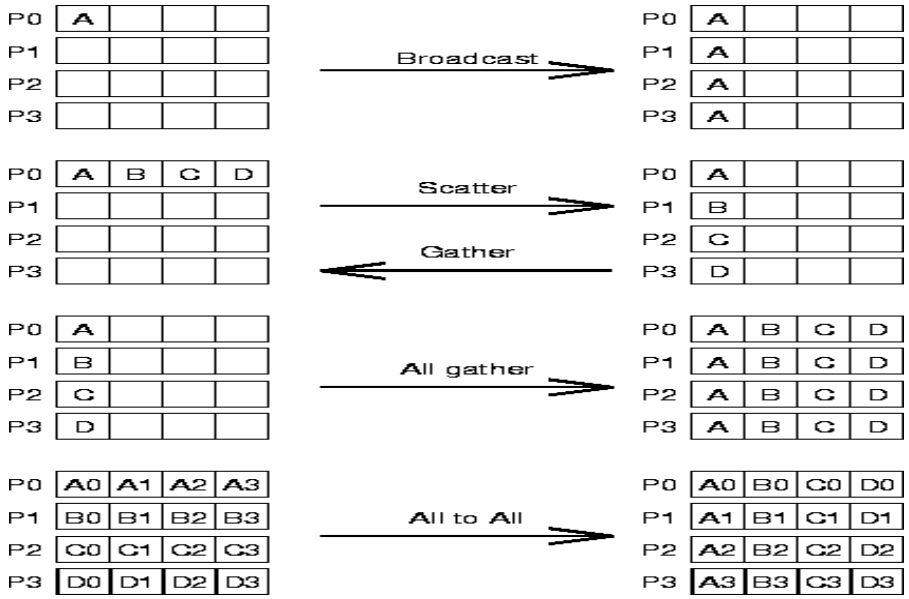
➢ 点对点通信

4 种消息传递函数	阻塞发送	MPI_Send(buffer,count,type,dest,tag,comm)
	非阻塞发送	MPI_Isend(buffer,count,type,dest,tag,comm,request)
	阻塞接收	MPI_Recv(buffer,count,type,source,tag,comm,status)
	非阻塞接收	MPI_Irecv(buffer,count,type,source,tag,comm,request)

4种通信模式	标准通信模式(MPI_SEND)
	缓存通信模式(MPI_BSEND)
	同步通信模式(MPI_SSEND)
	就绪通信模式(MPI_RSEND)

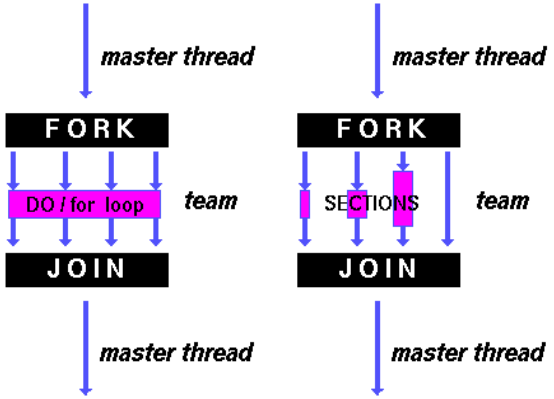
集合通信

- 通信:
  - MPI_Bcast(广播)
  - MPI_Scatter(发布)
  - MPI_Gather(搜集)
- 同步：MPI_Barrier
- 规约：MPI_Reduce



OpenMPI

设计目标	(1)提供一个标准的共享内存并行架构。 (2)建立一个简单的标示符集合，使得通过3、4个标示符就可以将程序并行化。 (3)能够同时提供粗粒度和细粒度并行。 (4)不需要编程人员学习消息通信包和撰写底层消息通信代码。
编程及执行流程	(1)基于OpenMPI思想编写并行程序； (2)利用OpenMPI编译器执行并行程序，生成多线程程序（包含线程、共享变量、子线程变量、barriers等）； (3)在多核计算机上执行编译后的并行程序。



支持for循环和sections并行

```
//java版OpenMPI框架jomp示例代码
    public static void main(String[] args) throws Exception {
        long t1 = System.currentTimeMillis();
        int n=2000000001;
        double sum=0;
        //omp parallel shared(n) private(i,sum)
        {
            //omp for reduction(+:sum)
            for(int i=0;i<n;i++){
                sum+=i;
            }
        }
        System.out.println("t="+(System.currentTimeMillis()-t1)+" : sum="+sum)
    }
```

缺点：

(1)需要预编译并行程序，流程复杂；

(2)编译后的并行代码可读性较差。

其它并行计算思想或平台缺点

- Map-Reduce（**Fork/Join**）

  需要对程序进行map、reduce设计，对串行代码调整较大；由于是分布式并行，因此仅适合比较容易map、reduce设计的场景，适用算法有限。

- Terracotta集群平台

  虚拟机层面并行，基本无需调整串行代码，但调度和负载均衡机制考虑较少。

议题

- 超大规模数据挖掘架构
  - 主流并行计算架构
  - 数据挖掘并行特点
  - GDM并行架构
- 数据分析方法论

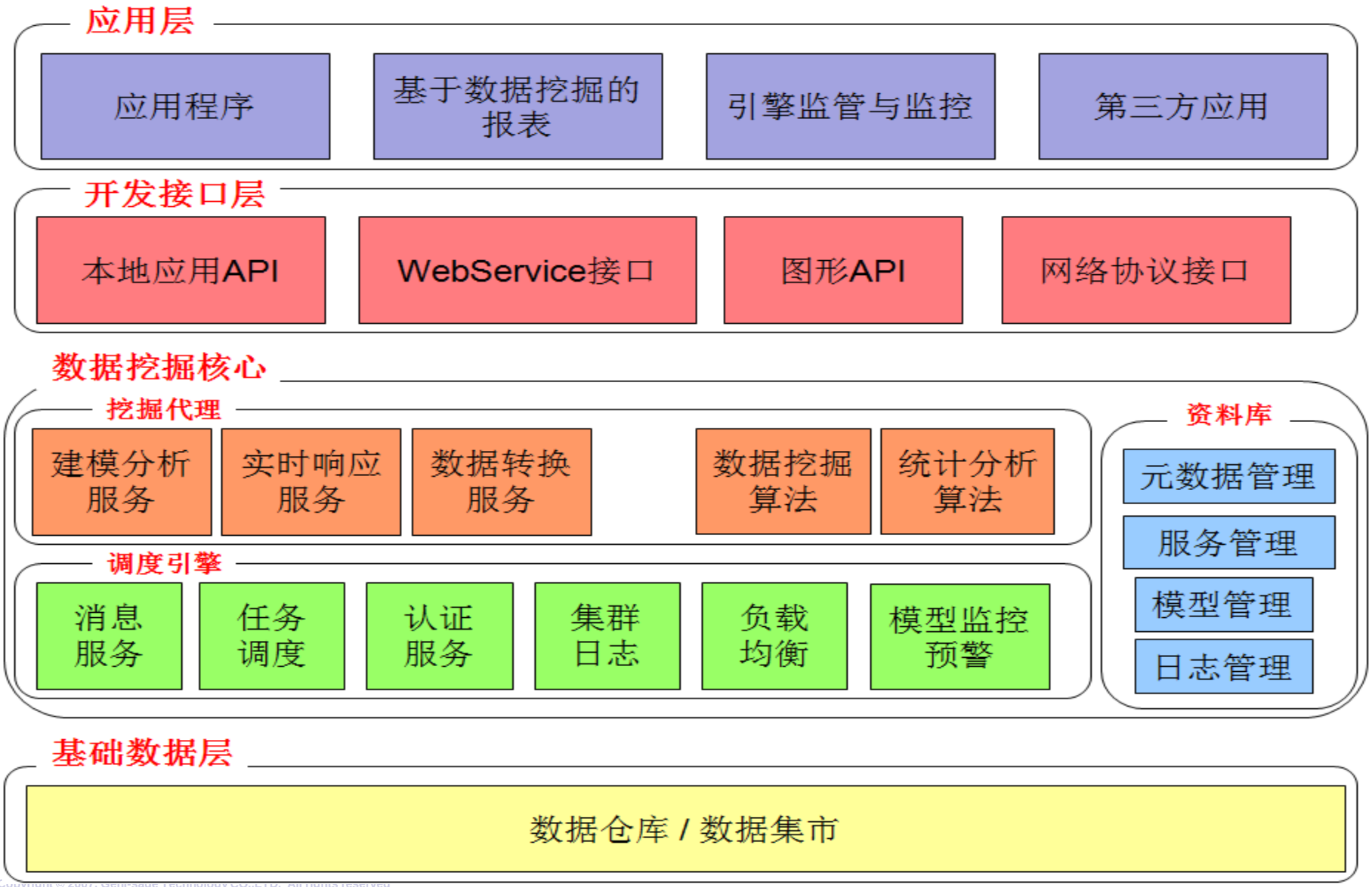
不同数据挖掘任务特点

任务类型	任务特点	数据挖掘举例	类比
分布式串行	耗时、 难以分解	决策树	
分布式并行	耗时、易分解、 不需要频繁通信、 同步性要求不高	不考虑属性之间相关性的属性重要性打分算法	银行业务办理柜台——目标：解决完当天所有客服问题。
集群并行	耗时、易分解、 需要频繁通信、 同步性要求较高	k均值	很多专家共同讨论、攻关一个难题——目标：攻关该难题。
网格实时任务	不耗时、 独立任务、 数据传输少、 计算资源消耗小	一个或多个模型的实时打分	

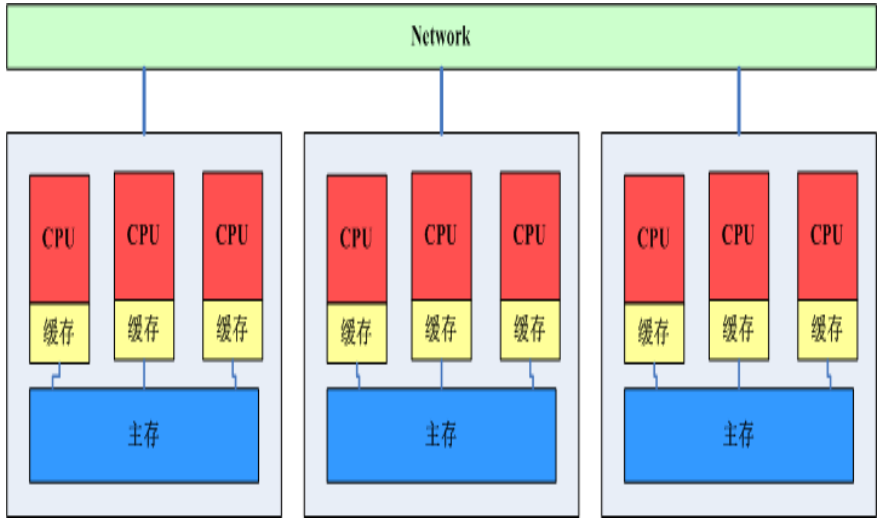
议题

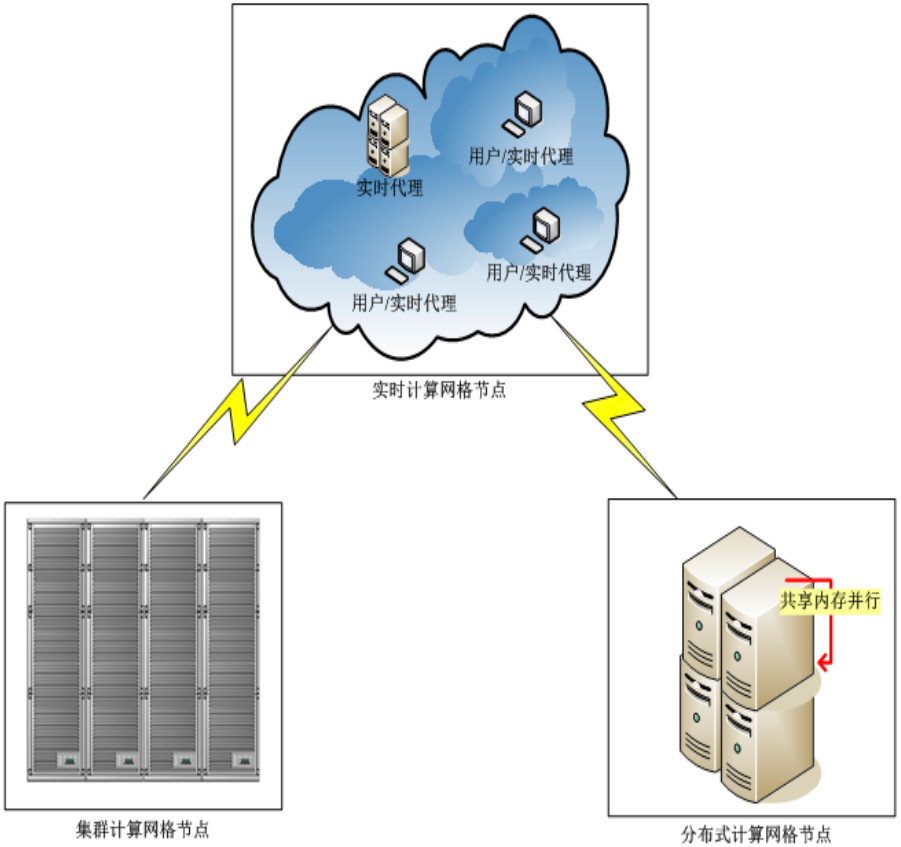
- 超大规模数据挖掘架构
  - 主流并行计算架构
  - 数据挖掘并行特点
  - GDM并行架构
- 数据分析方法论

GDM架构分层

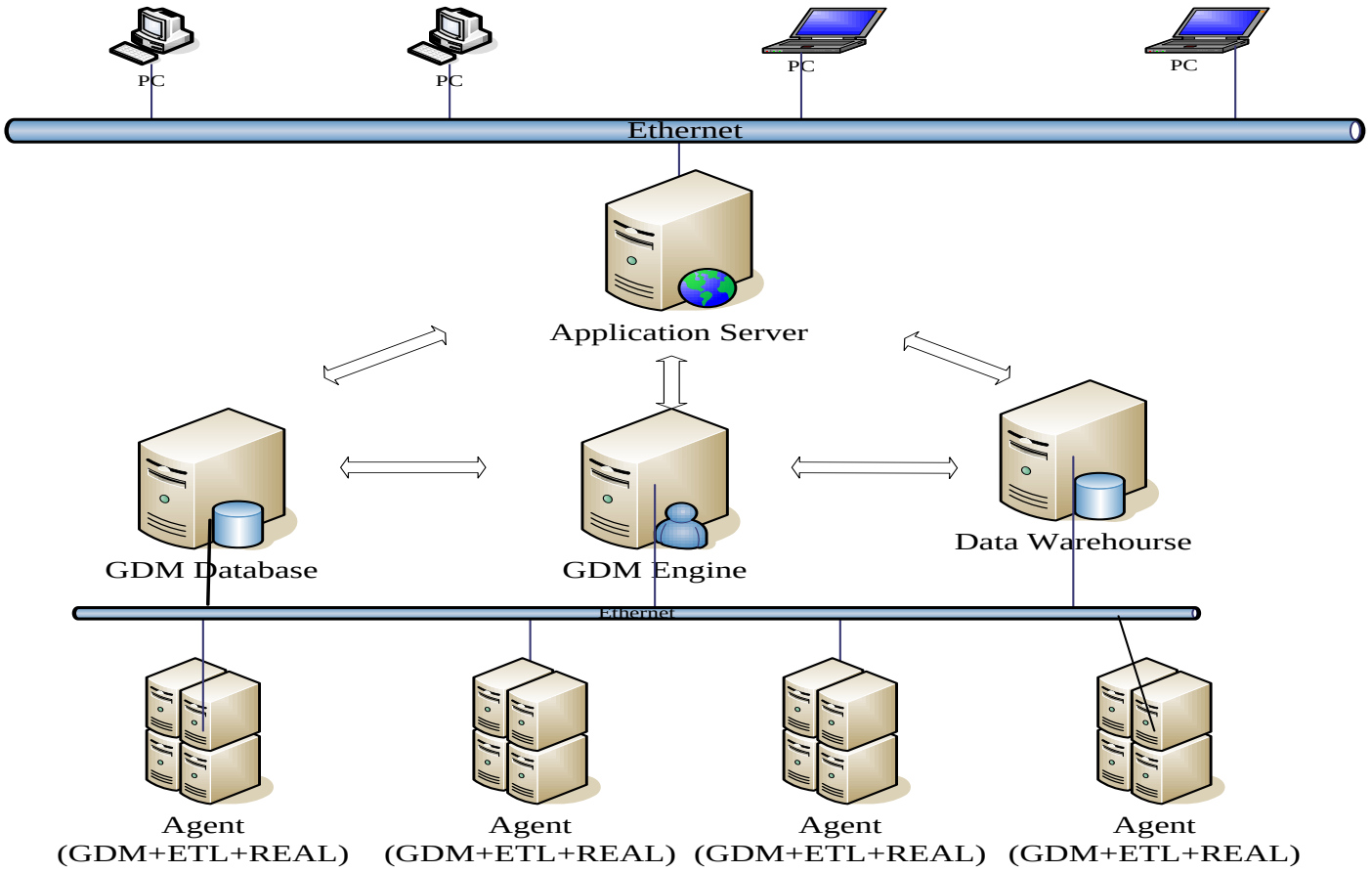


混合并行框架

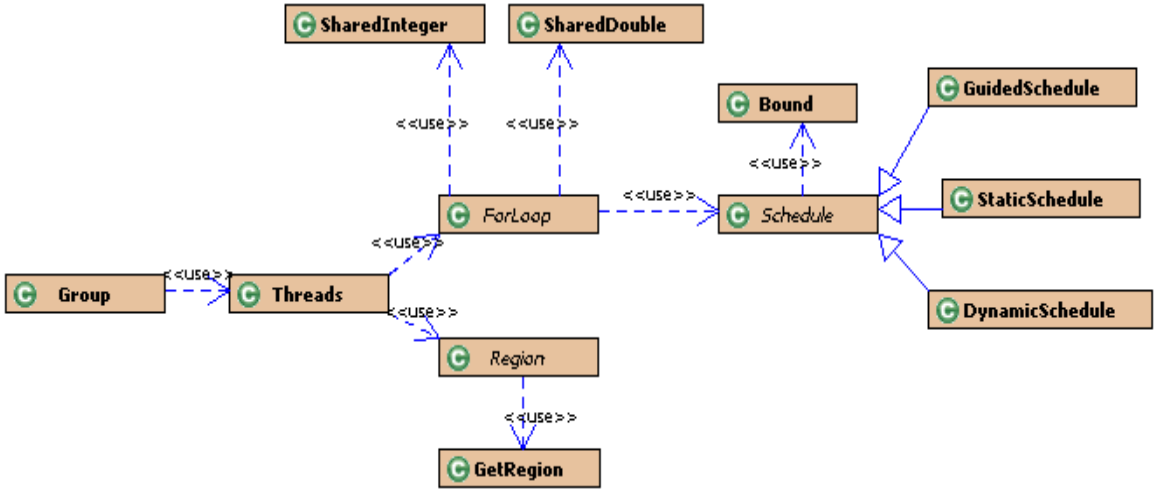




部署架构

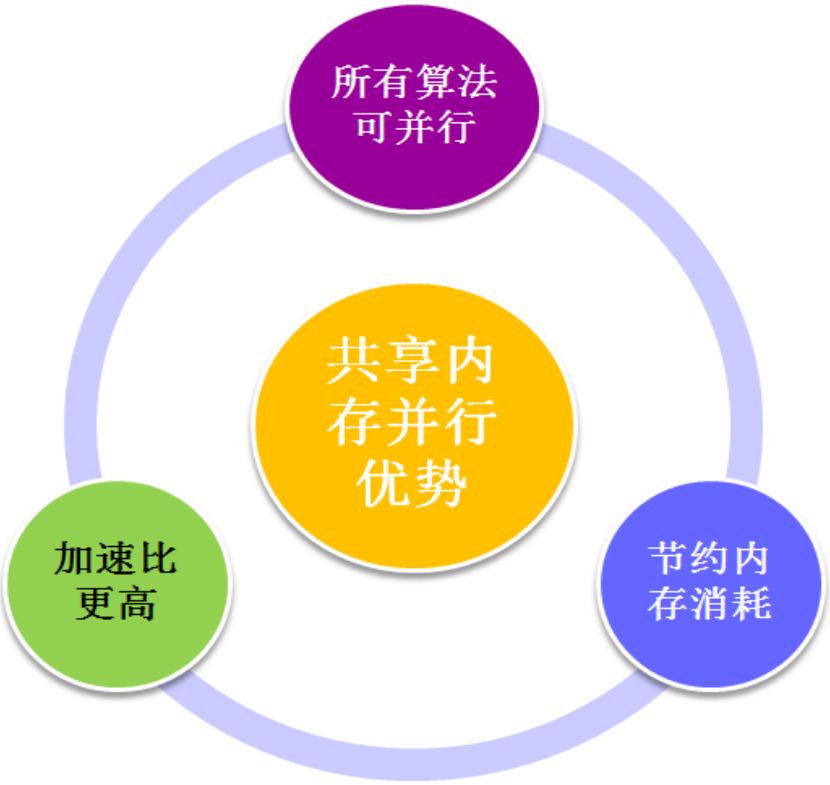


共享内存并行设计

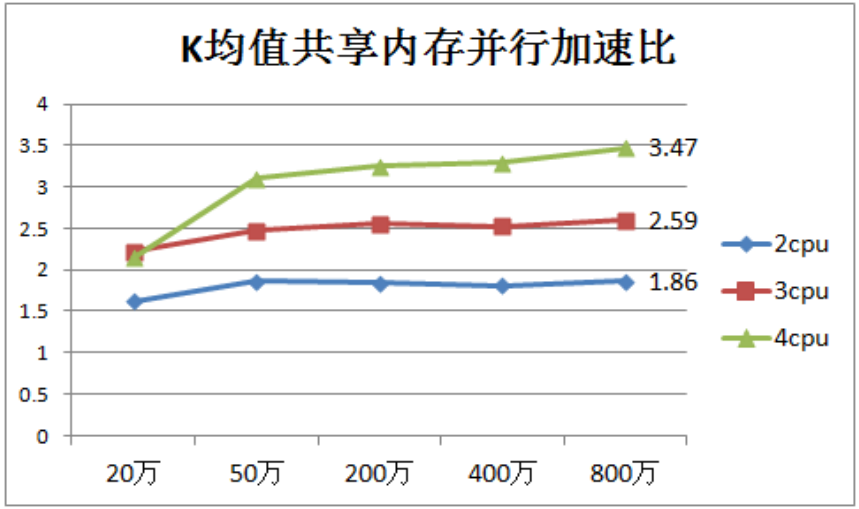



```
Group group = new Group(2)
group.execute(new ForLoop(low, up, Schedule.fixed()) {
      public void run(int start, int end) {
          此处定义线程自己的变量
          for (long i = start; i <= end; i++) {
              此处执行For循环的核心内容，与串行一致
          }
          此处将串行结果汇总到全局变量
        }
  }, true);
```

共享内存 vs 分布式并行



k均值并行内存消耗(M)			
	共享内存	分布式并行	内存节省
数据加载	634	400	
		445	
建模增量	25	23	
		23	
峰值	659	891	35.20%



任务监管及调度

议题

- 超大规模数据挖掘架构
- 数据分析方法论
  - 主流数据挖掘方法论
  - GDM方法论

议题

- 超大规模数据挖掘架构
- 数据分析方法论
  - 主流数据挖掘方法论
  - GDM方法论

主流数据挖掘方法论

- SPSS的5A(Assess-Access-Analyze-Act-Automate)：强调数据挖掘工具应具有的功能和能力
- SAS的SEMMA (Sample-Explore-Modify-Model-Assess)：强调基于SAS的分析步骤
- CRISP-DM(业务理解、数据理解、数据准备、模型建立、模型评估、应用部署)：强调方法论，独立于挖掘工具

GDM方法论

- 类似CRISP-DM，强调闭环的方法论
- 增加模型的监控，以及监控不达标时模型的改进；
- 强调模型评估不理想后，需要采取调整业务目标、进一步进行数据处理、重新选择建模算法、重新进行算法参数调整4种策略以改进模型；

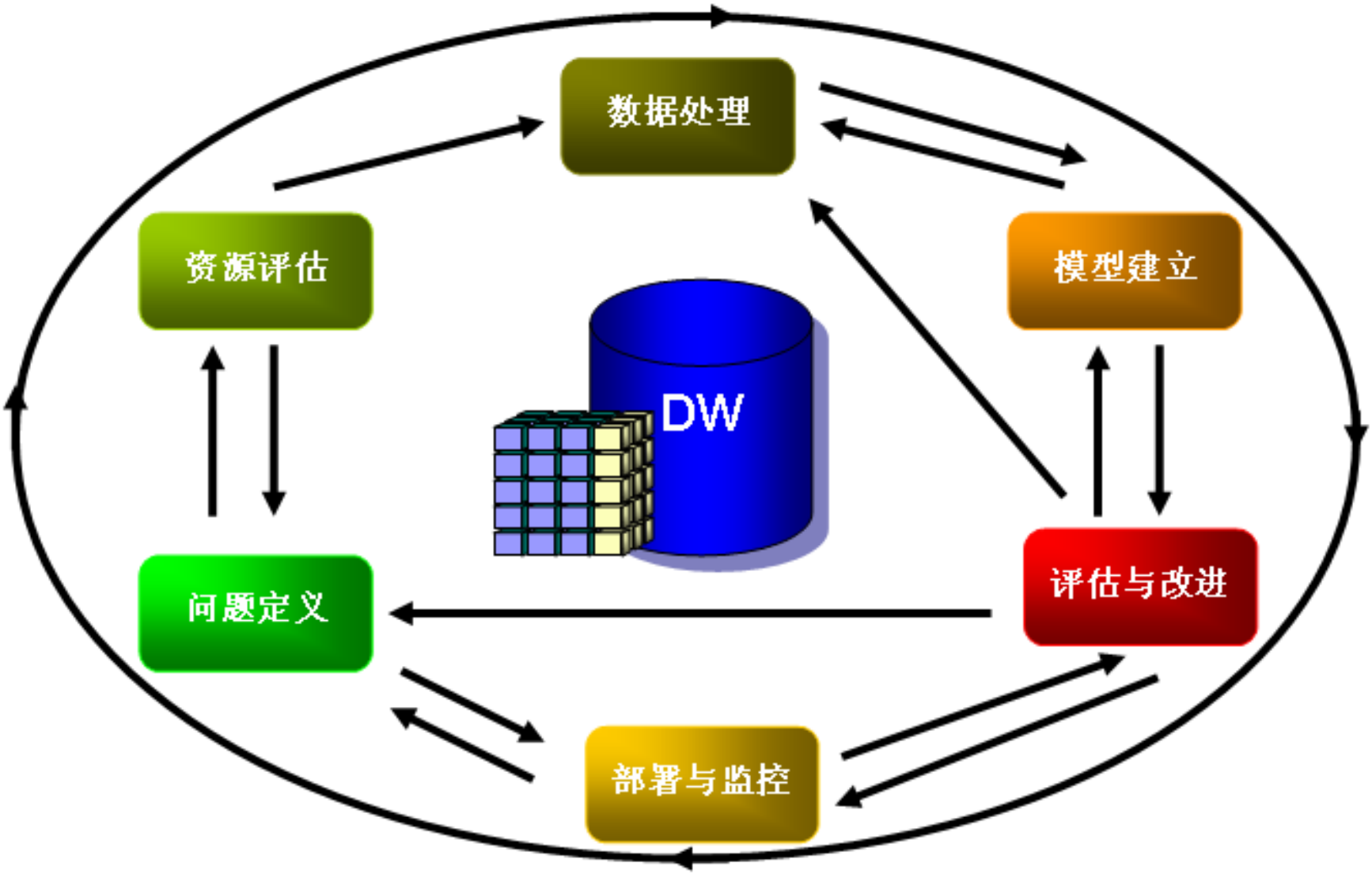
GDM方法论

6西格玛理论
- 设计管理DMADV：
  定义、评估、分析、设计、确认
- 改进管理DMAIC：
  定义、评估、分析、改进、控制

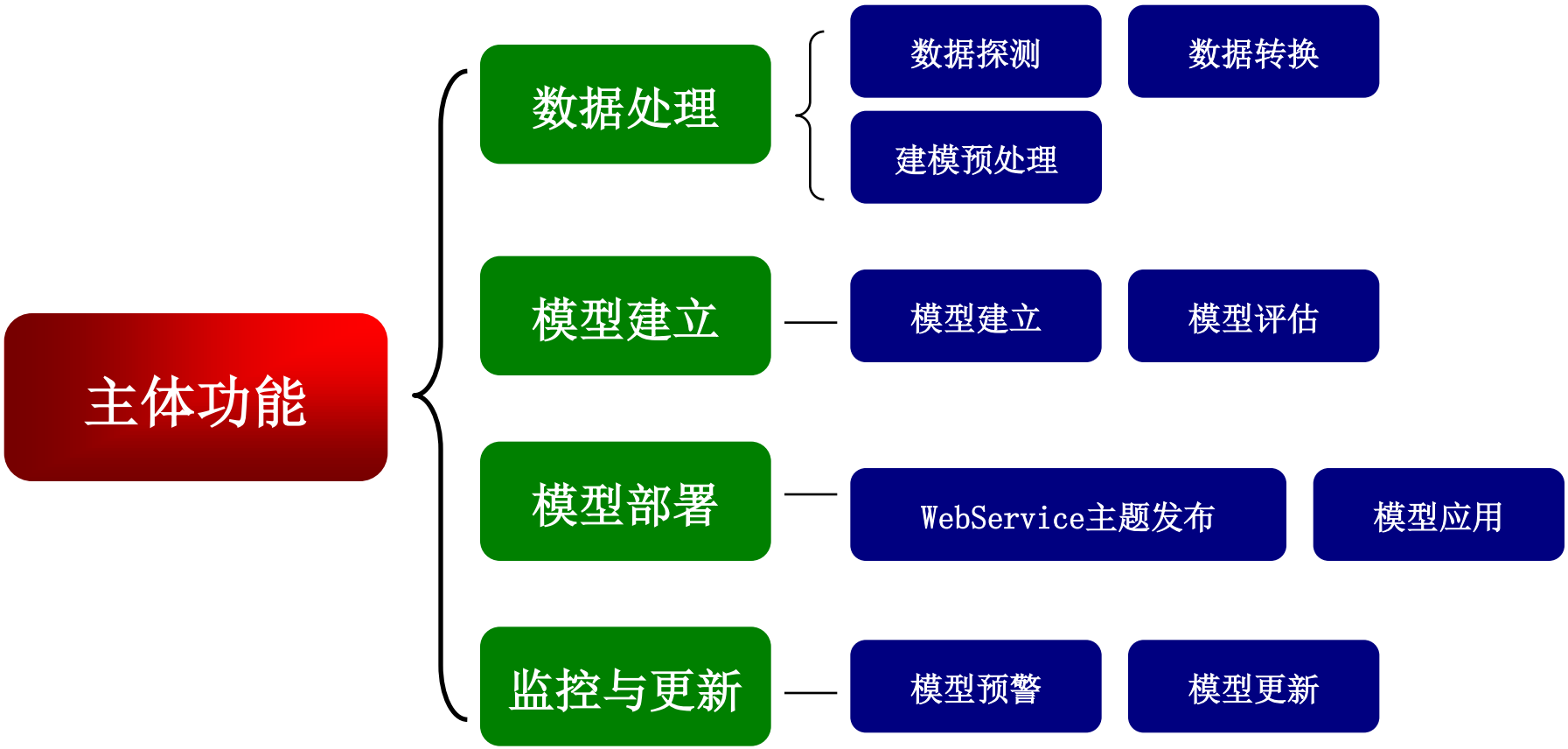
DMADIC：定义、评估、分析、设计、改进、控制

GDM数据挖掘流程：

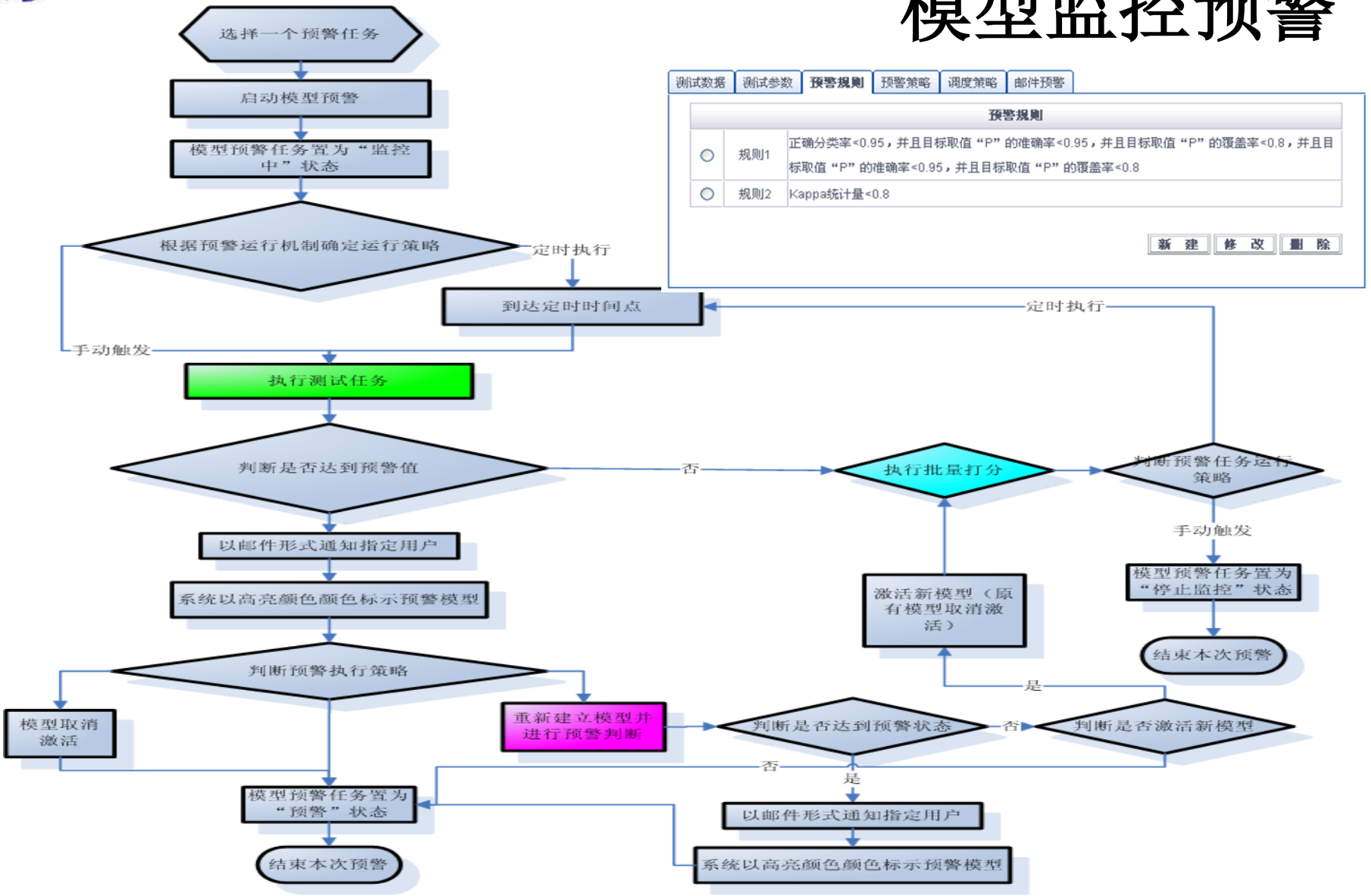
问题定义、资源评估、数据处理、模型建立、评估与改进、部署与监控



闭环流程对应系统主体功能



模型监控预警



Thank you

天津天才博通科技有限公司
电话： +86-22-60615379
传真： +86-22-58815089
网址：http://www.geni-sage.com
电子邮件：info@geni-sage.com

Further Insight,
Better Decisions