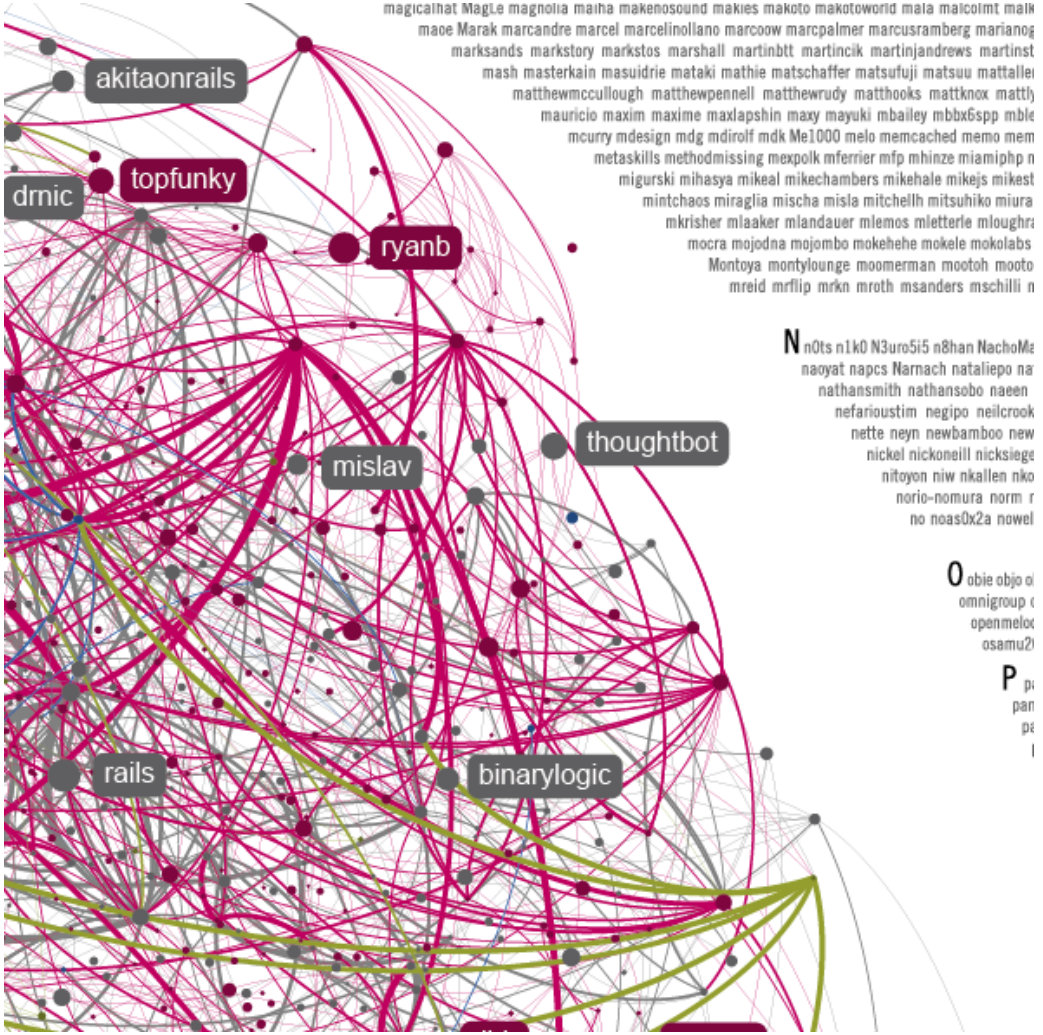
DATAGURU 炼数成金

DB2设计与性能优化

第2周



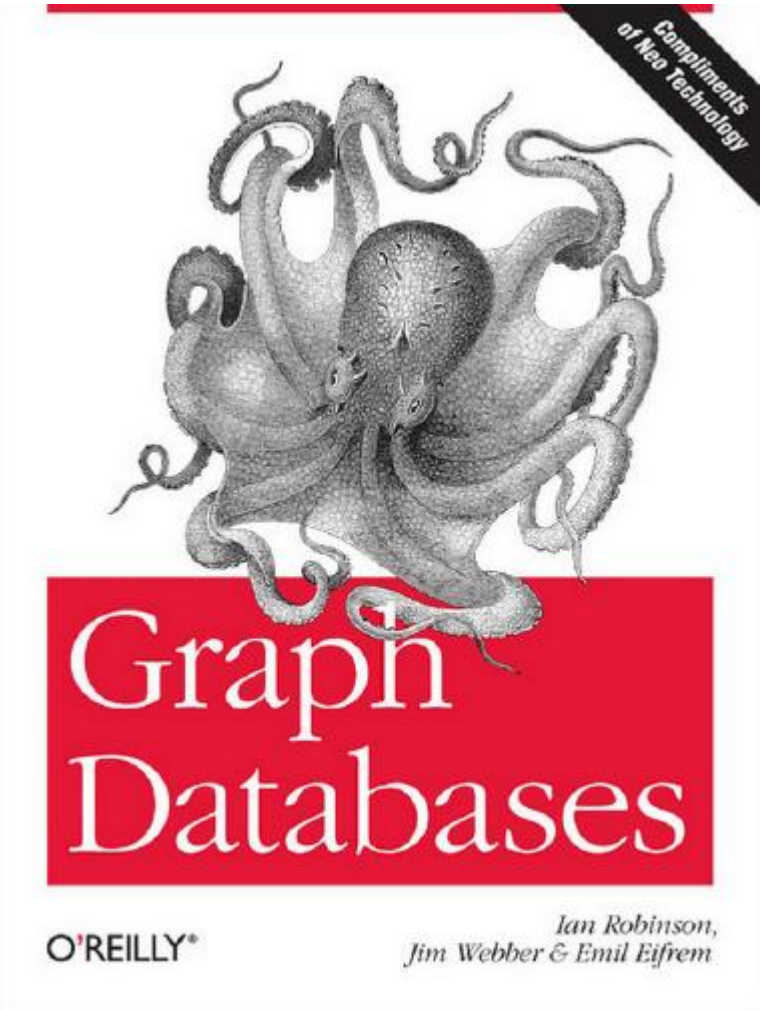
法律声明

【声明】本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

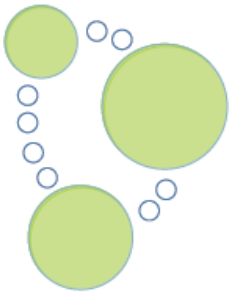
课程详情访问炼数成金培训网站

http://edu.dataguru.cn

图数据库参考书



The Neo4j Manual



neo technology
value in relationships

本周内容

- 教你三招
- 什么是性能问题
- 找到性能瓶颈
- 硬件规划
- 实战案例分享

第一招：解决硬解析的利器-绑定变量

背景：

绑定变量是解决动态语句硬解析的利器，能解决OLTP系统中Package cache的过度耗用以提高性能。

用法：

```
//激活语句集中器
  db2 update db cfg using STMT_CONC LITERALS
//下面的JAVA代码使用绑定变量，避免对动态语句硬解析
  PreparedStatement p = conn.prepareStatement
                ("SELECT name FROM emp WHERE id = ? AND dept = ?");
  p.setInt(1, 314159);
  p.setString(2, "SALES")
```

使用场合：

在OLTP环境中SQL语句重复执行频度高，但处理的数据量较少，结果集也相对较小，解析时间通常会接近或高于执行时间，因此该场合适合使用绑定变量。

第二招：从数据库到应用-行预取

背景：

有时候，我们发现一个需要返回大量结果集的查询语句性能很差，但是却不是数据库引擎导致的，而是应用导致的，怎么解决呢？

用法：

```
//方法1：
connProp = new Properties();
connProp.put("defaultRowPrefetch", 100)
dataSource.setConnectionProperties(connProp);
//方法2
sql = "select id, name from t"
statement = connection.prepareStatement(sql);
statement.setFetchSize(100);
resultset = statement.executeQuery();
```

使用场合：

应用程序请求驱动从数据库返回记录的时候，会读取多条满足条件的记录并存储在客户端的内存中，这样后续的请求可以从客户端内存中直接去读

第三招：从应用到数据库-批量提交

背景：

有时候，我们需要插入大量记录到数据库，如果逐条逐条的插入，则性能低下，这个也不是数据库引擎导致的，应用需要优化，那么如何解决呢？

用法：

```
sql = “insert into t values(?,?)”
statement = connection.prepareStatement(sql);
for(int i=1; i<100000;i++)
{
statement.setInt(1,i);
statement.setString(2,”hello…”);
statement.addBatch();
}
statement.executeBatch();
statement.close();
```

使用场合：

有批量作业需要处理的场合，能大幅提升性能。

Agenda

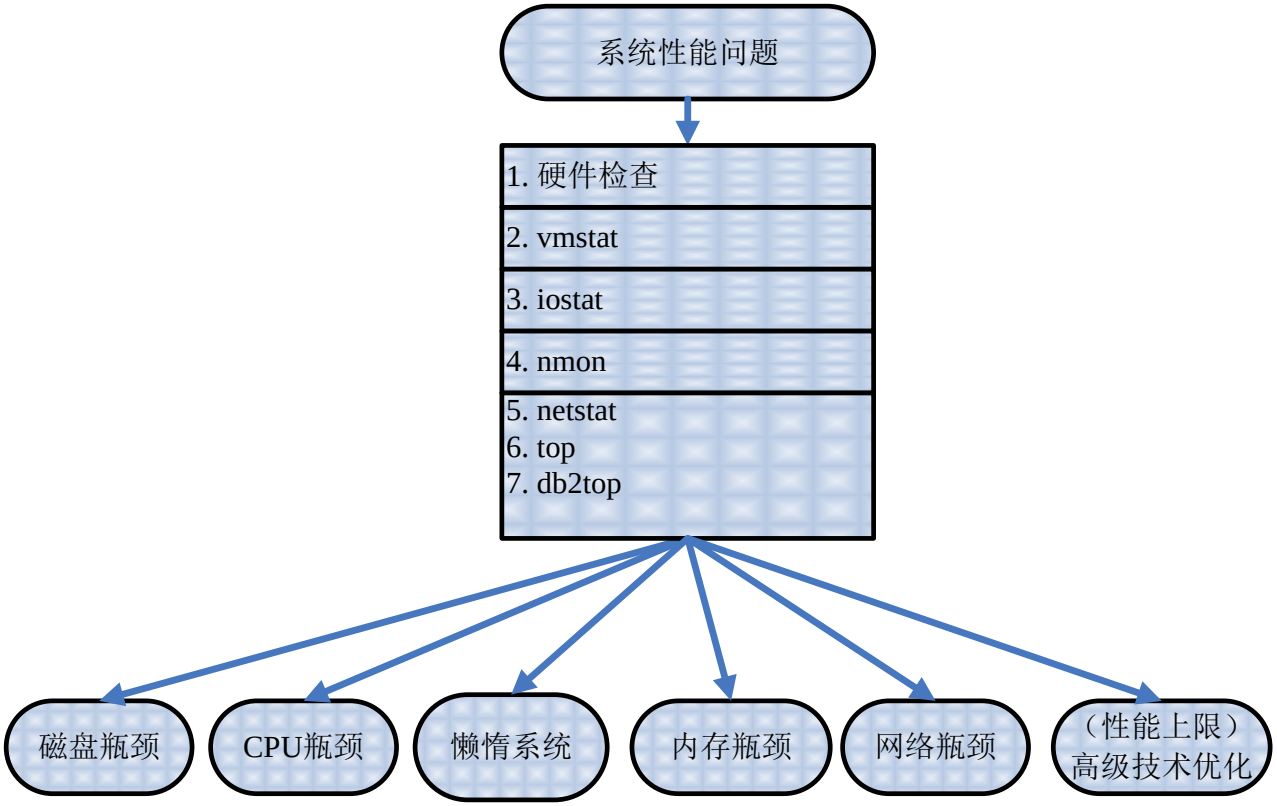
- 教你三招
- 什么是性能问题
- 找到性能瓶颈
- 硬件规划
- 实战案例分享

什么是DB2性能问题?

- 性能问题比功能问题难
  - 性能变慢，出现大量的锁超时，原因竟然是统计信息过时，全表扫描导致
  - 性能问题有时候是间歇性的
  - 性能问题是可以避免的

- 解决性能问题的通常办法（假如是新手）
  - 慌乱
  - 随机性调整，靠运气
  - 买更多的硬件（CPU、内存、磁盘等）
  - 指责DB2...
    - AIX / Windows / Linux...
    - IBM / HP / Sun /...

六种类型的瓶颈


系统性能问题
1. 硬件检查
2. vmstat
3. iostat
4. nmon
5. netstat
6. top
7. db2top
磁盘瓶颈
CPU瓶颈
懒惰系统
内存瓶颈
网络瓶颈
（性能上限）
高级技术优化

度量性能的三个指标

- 响应时间:

  数据库服务器收到应用请求后完成处理并返回给应用总共所消耗的时间， 它反映了服务器的处理速度。

- 事务吞吐量:

  通常用每分钟处理的事务数（TPM）来计算，这个指标反映了系统的事务处理能力。

- 资源利用率:

  数据库服务器在处理事务或者查询的过程中，系统资源包括CPU、I/O以及磁盘的使用情况，这个指标反映了系统资源是否被服务器有效利用。

Agenda

- 教你三招
- 什么是性能问题
- 找到性能瓶颈
- 硬件规划
- 实战案例分享

制定优化目标

- 根据成本效益分析（Cost-Benefit Analysis）模型，付出的努力永远要和获得的收益取得平衡。
- 优化无止境，因为没有目标就不知道何时停止研究更好的解决方案。因为性能是有时间和金钱成本的。
- 当DBA考虑需要多少时间和金钱用于性能优化时，要评估一下所花费的时间成本和金钱成本。

测试的方法：性能基准测试（Benchmarking）

- 涉及开发和运维整个数据生命周期
- 基于可控条件
  - 重复运行有性能问题的应用
  - 迭代之间，更换条件:
    - 参数配置
    - SQL语句
    - 索引
    - 表空间配置
    - 硬件配置
  - 重复，直到整个应用性能满足要求
- 性能基准测试的要求：
  - 测试可重复
  - 每次迭代启动时，系统状态相同
  - 没有其他应用干扰
  - 软硬件和生产环境基本一致

分析的方法：解决性能问题的四个步骤

- 第一步，问题监控：监控硬件和数据库
- 第二步，配置检查：硬件配置、操作系统配置和数据库配置
- 第三步，设计检查：逻辑设计和物理设计的正确性
- 第四步，性能优化：考虑投入产出比，寻找合适的解决方案

性能调优的限制

- 多少时间和金钱可以被投入?
  - 考虑投资回报率
- 调优可以解决的问题
  - 响应时间慢的问题
  - 事务吞吐低的问题
- 更好的解决方案是从架构着手
  - 更快更多的存储
  - 更快更多的CPU
  - 更多的内存
  - 更快的网络

Agenda

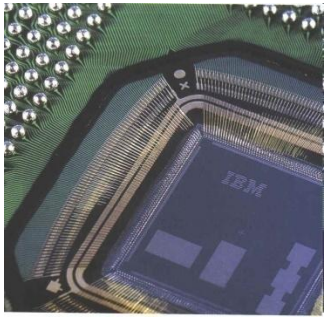
- 教你三招
- 什么是性能问题
- 找到性能瓶颈
- 硬件规划
- 实战案例分享

CPU

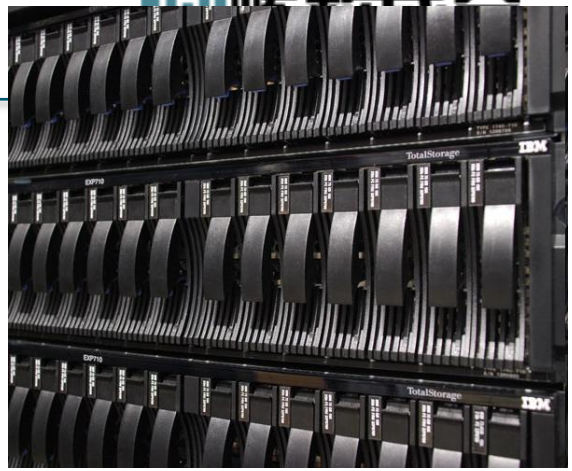
- OLAP/BI应用:
  - 每处理器内核支撑200-300 GB活动数据
- OLTP:

浏览www.tpc.org TPC-C DB2 9 测试结果

  - 8 core IBM POWER5 1.9 GHz = ~430K TPM
  - 8 core IBM POWER6 4.2 GHz = ~630K TPM
  - 64 core IBM POWER6 5.0 GHz = ~6M TPM
  - 8 core Intel Xeon 3.33 GHz = ~314K TPM
  - 16 core Intel Xeon 2.93 GHz = ~517K TPM



存储



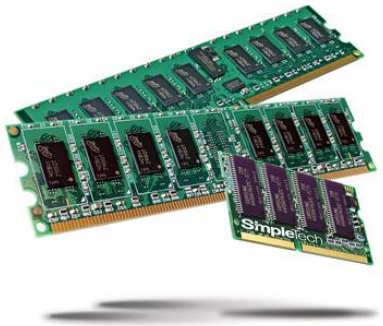
考虑因素:

1. I/O吞吐
   - I/Os per Second (IOPS)
   - Megabytes per Second (MPS)
2. 容量
3. 单独的日志空间

经验公式:

- 15K RPM drive = ~150-175 IOPS @ 7-8 milliseconds response time
- 10 to 20 disk per core
  - 10 for Intel/AMD (System x)
  - 20 for POWER5 and POWER6 (System p)

内存



- CPU和磁盘之间的缓冲
- 受共享内存的限制
  - 32 bit ~4GBs (Windows /Linux)
  - 64 bit ~无限制
- 4-8GB RAM per core
  - 4GB per core for Intel/AMD (System x)
  - 8GB per core for POWER5 and POWER6 (System p)

Agenda

- 教你三招
- 什么是性能问题
- 找到性能瓶颈
- 硬件规划
- 实战案例分享

索引优化-加强治理

背景：

在客户环境中，我们通常通过在表上建各种索引来提高查询速度，但索引过多过滥严重影响了插入和更新时的性能和存储开销，如何选择合适的列作为索引成了用户比较头疼的问题。

案例：

某电信公司，一个表上建立了包含3个列(A, B, C)的索引，如果(A, C)或者(B, C)这样的组合查询条件很多情况下是无法利用(A, B, C)索引的，那么怎么提升他们的查询速度呢？

解决办法：

1，方法一：单独为(A, C)和(B, C)建立索引，这导致索引过多过滥。

2，方法二：利用DB2 V10提供的Jump Scan技术，无需建立额外索引，(A, C)和(B, C)这样的查询可以直接用上（A,B,C）上的索引。

大表优化-启用压缩

背景：

在生产环境中，经常面临数据快速增长的问题，为了减少存储空间，我们通过对大表或者流水表进行压缩来减少存储空间。

案例：

某银行，有一个非常大的流水表，目前已近有2600万行记录，每天增长100万条，数据量增长非常快，按照目前的增长速度，再过一段时间就会把表空间占满。

解决办法：

1，方法一：使用DB2 v9.7提供的表压缩技术对其进行压缩，取得了一定效果，但是还不够。

2，方法二：利用DB2 V10提供的自适应压缩（Adaptive Compression）技术，再次对底层的数据页进行压缩，这使得在表压缩的基础上，数据再压缩40% - 50%。

热表优化-将表按照访问频度分组（1）

背景：

数据库存在“二八法则”，20%的表往往是被80%的用户经常访问，80%的表被20%的用户经常访问。针对这种不同访问频度的表，需要有针对性的优化方案。

案例：

某券商在某个月，某些持仓大户的数据需要频繁读取并且对IO性能要求高；某些持仓散户的数据需要一般性的读取；其他散户则不怎么读取。用户数据的访问频度不固定，下个月可能就会发生变化了。

解决办法：

1，方法一：为不同读取频度的用户数据，创建不同的表空间，数据访问越频繁，容器就建在越快速的I/O设备上。如果发生变化，DBA需要花费很多精力在不同表空间之间移动数据。

2，方法二：（下一页）

热表优化-将表按照访问频度分组（2）

解决办法：

方法二： DB2 Galileo提供了存储组的特性：

- 创建存储组（ Storage Group ）

db2 create stogroup sg_hot on '/ssdisk'；

db2 create stogroup sg_warm on '/raidisk'；

db2 create stogroup sg_cold on '/normaldisk'；

- 创建表空间

db2 create tablespace tbshot using stogroup sg_hot；

db2 create tablespace tbswarm using stogroup sg_warm；

db2 create tablespace tbscold using coldgroup sg_cold；

- 表空间在线更改（从sg_warm->sg_hot,sg_hot->sg_warm）

db2 alter tablespace tbswarm using stogroup sg_hot；

db2 alter tablespace tbswarm using stogroup sg_warm；

数据仓库优化-提升实时性

背景：

目前国内数据仓库进行分析的数据都是从生产系统抽取-转换-加载（ETL）而来，由于这些操作必须在维护窗口执行，所以无法做到实时，通常都会滞后1天，也就是说是T+1的。

案例：

某电信公司， 使用ETL工具将业务数据加载到基于DB2的经分系统中，由于数据越来越大，在维护窗口内无法完成所有数据的加载工作，导致从T+1变成了T+1.5

解决办法：

1，方法一：使用CDC进行24小时实时复制，不需要维护窗口，但是实施复杂。

2，方法二：利用DB2 V10提供的CDI (Continous Data Ingest) 工具，与ETL工具相比，CDI运行时，目标表是可用的，加载数据不需要维护时间窗口。可以对数据进行格式转换，完成简单的ETL功能。还可以从上一次失败的地方重新开始加载数据。

Thank You

炼数成金逆向收费式网络课程

- Dataguru（炼数成金）是专业数据分析网站，提供教育，媒体，内容，社区，出版，数据分析业务等服务。我们的课程采用新兴的互联网教育形式，独创地发展了逆向收费式网络培训课程模式。既继承传统教育重学习氛围，重竞争压力的特点，同时又发挥互联网的威力打破时空限制，把天南地北志同道合的朋友组织在一起交流学习，使到原先孤立的学习个体组合成有组织的探索力量。并且把原先动辄成千上万的学习成本，直线下降至百元范围，造福大众。我们的目标是：低成本传播高价值知识，构架中国第一的网上知识流转阵地。
- 关于逆向收费式网络的详情，请看我们的培训网站 http://edu.dataguru.cn

Thanks

FAQ时间