友乾营

专注数据技术的社群

并行取数提升数据读取性能

15 期

www.raqsoft.com.cn/yqy

数据库的JDBC到底有多慢?

友乾营

【测试用例】

从3000万行，8列的customer表（文本大小4.9GB），分别从Oracle和MySQL中取数测试

测试结果（时间单位：秒）

	第一次	第二次	每秒行数
Oracle	293	281	106000
MySQL	518	381	79000

硬件环境：2个Intel2670 CPU，主频2.6G，共16核，内存64G，SSD硬盘

3000万行的表要读 5分钟 ！！！ OMG

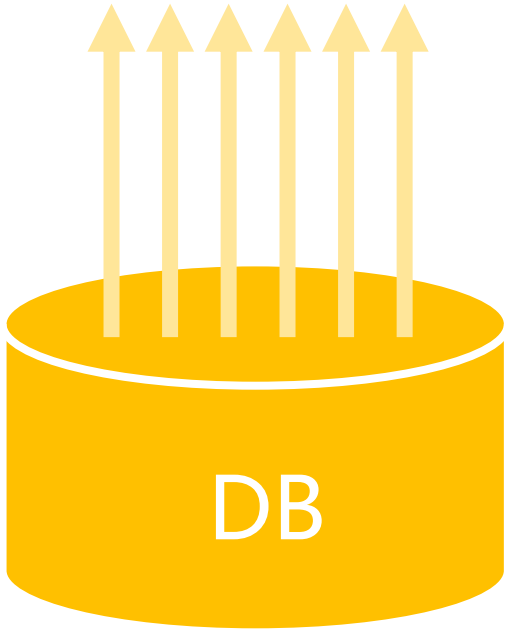
如何加速?

友乾营

通过并行取数加速读取效率，利用多CPU能力

但是，JAVA实现并行程序太难（要考虑资源共享冲突等麻烦事务）

有没有简单的办法?



单表并行

1. 如何分段?
   a.有索引分段
   b.无索引分段
2. 外存情况

并行数设置

1. 不大于CPU核数
2. 远超过CPU核数

多表并行

并行取数与关联计算

利用索引字段分段

友乾营

均匀分段：单表并行取数前，需要进行数据分段，要保证每个分段的数据相对均匀

利用索引可以加快分段数据定位，快速实施并行取数

举例

并行读取指定时间范围的订单表数据

（索引字段：订单ID）

代码（1）-准确计算数据范围

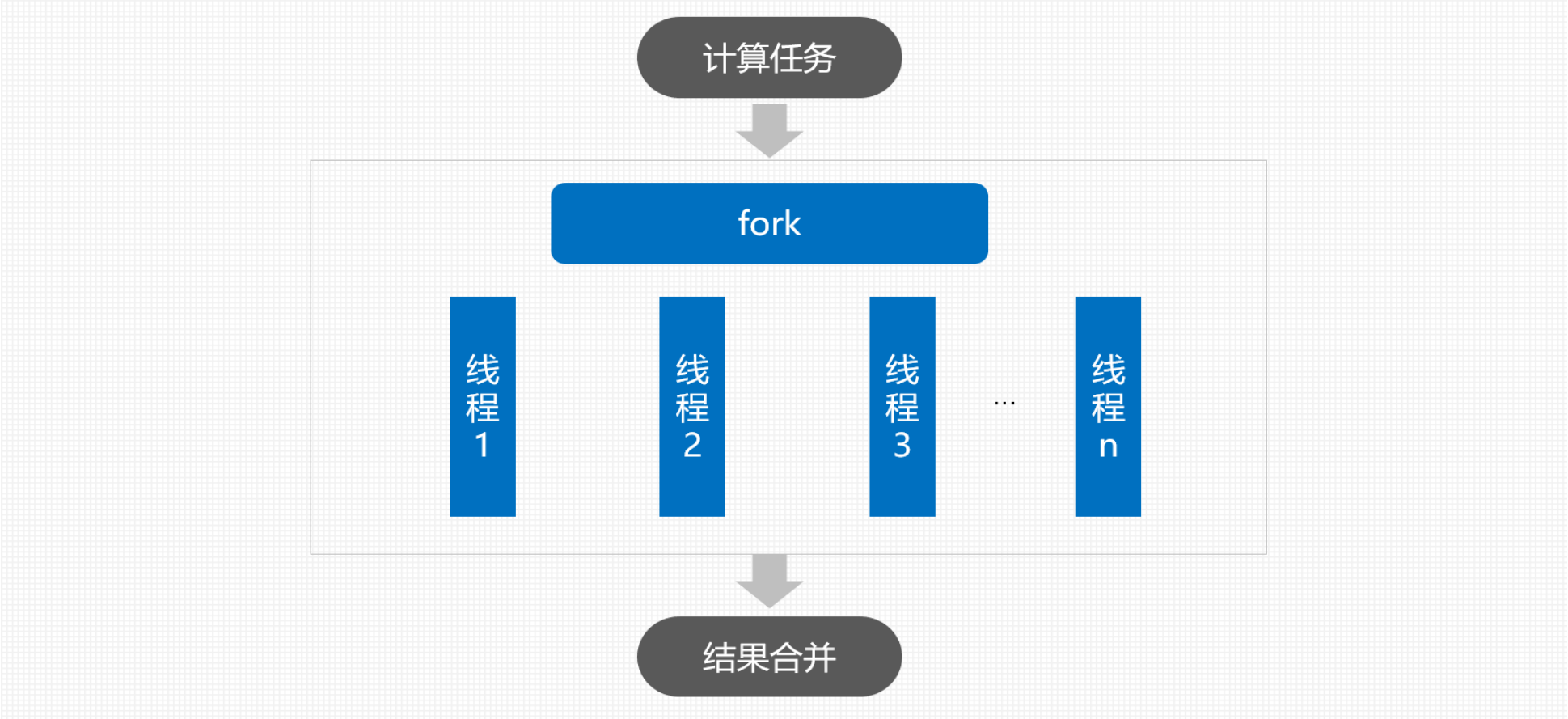
	A	B	C
1	=connect("db")		
2	=A1.query("select min(订单ID) 最小ID,max(订单ID) 最大ID from 订单 where 订购日期>=? and 订购日期<=?",begin,end)	=b=A2.最小ID	=e=A2.最大ID
3	=p=4	/并行数	=range(b,e+1,p)
4	=C3.to(,p)	/分段参数初值	
5	=C3.to(2,)	/分段参数终值	
6	fork A4,A5	=connect("db")	
7		=B7.query@x("select * from 订单 where 订单ID>=? and 订单ID<? and 订购日期>=? and 订购日期<=?",A6(1),A6(2),begin,end)	
8	=A6.conj()	/合并查询结果	

代码（2）-有效利用索引

	A	B	C
1	=connect("db")		
2	=A1.query("select min(订单ID) 最小ID,max(订单ID) 最大ID from 订单)	=b=A2.最小ID	=e=A2.最大ID
3	=p=4	/并行数	=range(b,e+1,p)
4	=C3.to(,p)	/分段参数初值	
5	=C3.to(2,)	/分段参数终值	
6	fork A4,A5	=connect("db")	
7		=B7.query@x("select * from 订单 where 订单ID>=? and 订单ID<? and 订购日期>=? and 订购日期<?",A6(1),A6(2),begin,end)	
8	=A6.conj()	/合并查询结果	

关于fork

在集算器中，通过fork语句可以启动多个线程实施并行计算，而且集算器还提供了多种merge函数可以很方便合并并行结果



注意

友乾营

需要注意**数据库连接必须在并行线程中建立**，以便为多线程分别使用，若共用一个连接无法起到加速取数的效果，数据库会自动把同一连接上的多个请求改为串行执行。因此只有当数据库负担不重，有足够多连接可用时才可以使用并行取数提升性能

性能比较

测试用例：基于订单表，读取指定时间范围的订单数据

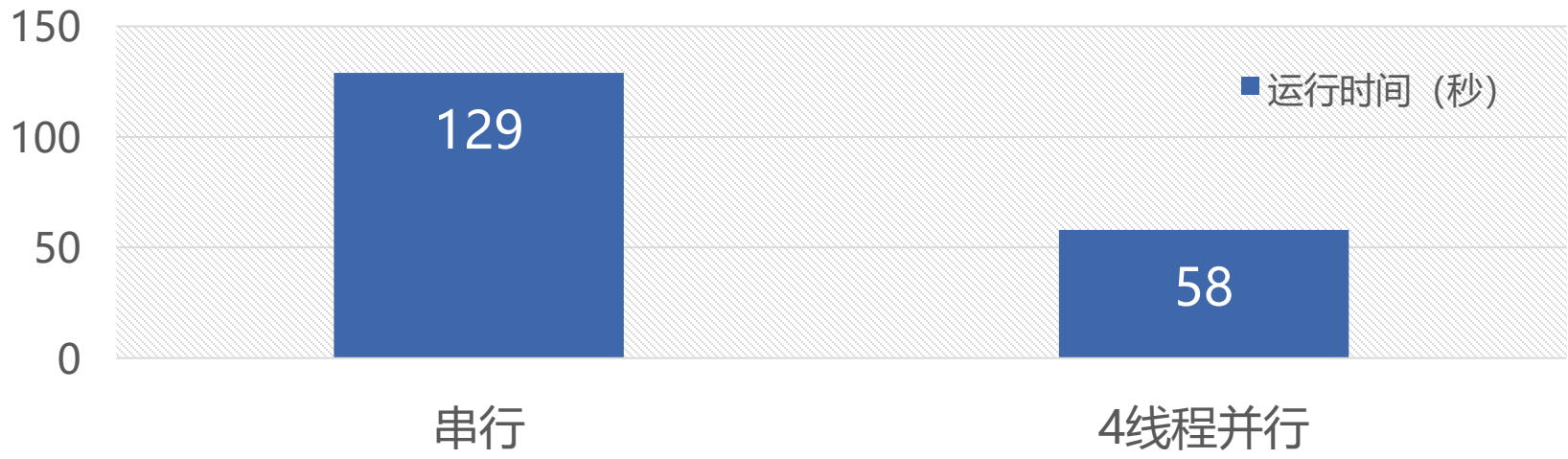
测试环境（PC）：

处理器:	Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz 2.40 GHz
已安装的内存(RAM):	8.00 GB (7.90 GB 可用)
系统类型:	64 位操作系统，基于 x64 的处理器

数据库：oracle12c

订单表记录数：500万

读取数据范围：3年数据（共8年）



非索引字段分段

友乾营

如果数据库负担不重时，也可以基于非索引字段进行分段（如日期），相对JDBC取数时间，多次遍历库表时间也并不是很大

好处：不需要事先查询数据库以确定起止段界

举例

并行读取指定时间范围的订单表数据

代码

友乾营

	A	B	C
1	=connect("db")		
2	=range(begin,end+1,4)		
3	=p=4	/并行数	
4	=A2.to(,p)	/分段参数初值	
5	=A2.to(2,)	/分段参数终值	
6	fork A4,A5	=connect("db")	
7		=B6.query@x("select * from 订单 where 订购日期>=? and 订购日期<?",A6(1),A6(2))	
8	=A6.conj()	/合并查询结果	

外存情况

友乾营

有时某一条语句（一个表）的数据量较大，分段后并行子任务仍然无法全部加载到内存中，这时通过游标查询数据再合并

举例

并行读取指定时间范围的订单表数据

（数据规模巨大，分段后仍然无法全内存）

代码

友乾营

	A	B	C
1	=connect("db")		
2	=A1.query("select min(订单ID) 最小ID,max(订单ID) 最大ID from 订单 where 订购日期>=? and 订购日期<=?",begin,end)	=b=A2.最小ID	=e=A2.最大ID
3	=p=4	/并行数	=range(b,e+1,p)
4	=C3.to(,p)	/分段参数初值	
5	=C3.to(2,)	/分段参数终值	
6	fork A4,A5	=connect("db")	
7		=B6.cursor@x("select * from 订单 where 订单ID>=? and 订单ID<? and 订购日期>=? and 订购日期<=?",A6(1),A6(2),begin,end)	
8	=A6.mcursor()	/合并查询结果	
9	=file("\usr\订单.txt").export@t(A8)	/基于游标写入文件	

说明

基于外存游标并行查询与全内存方式非常类似，当内存资源较紧张时可以通过外存计算的方式减少内存占用

游标归并时，因为各个线程的运行速度无法保证规律性，所以基于多线程导出数据时次序不可控，对数据顺序有要求时不能使用这个方法

注意对结果顺序的要求

并行数设置

友乾营

通常并行程序的并行数量建议**不要超过CPU核数**（≤CPU核数）

因为更多的任务数并不会增加并行度，而且还可以避免CPU进行线程切换带来的额外时间开销

情况一

并行数≤CPU核数

将任务数设置到远大于CPU核数，可以设置为CPU核数的倍数个，这样多CPU负载也可以达到动态平衡，而且某些计算还可以简化分段

情况二

并行数 远大于 CPU核数

举例

并行读取一年内的订单表数据

只查询某一年数据就可以把线程数设置为12（月），从而简化分段

	A	B	C
1	fork to(1,12)	=connect("demo")	
2		=B1.query@x("select * from 订单 where month(订购日期)=? and 订购日期>=? and 订购日期<=?",A1,begin,end)	
3	=A1.conj()	/合并查询结果	

说明

集算器提供了多线程任务动态平衡机制，当任务数大于并行数配置时，集算器会自动为计算结束的线程分配下一个任务，这时可以保证某个线程会多跑几个小任务，另一个线程只跑少量大任务，达到总体平衡，而不必拘泥于必须把数据量平均分配

多表并行取数

友乾营

在一些多SQL查询场景（如报表多数据集）下仍然可以通过并行同时执行多条语句进行取数

订单表	客户表	雇员表	订单明细表	产品表
订单ID	**客户ID**	**雇员ID**	**订单ID**	**产品ID**
订购日期	客户名称	**形式**	**产品ID**	产品名称
客户ID	所在区域	回款日期	单价	类别
订单金额	…	回款金额	数量	…
…		…	…	

举例

并行读取5个表数据，并完成关联

代码

友乾营

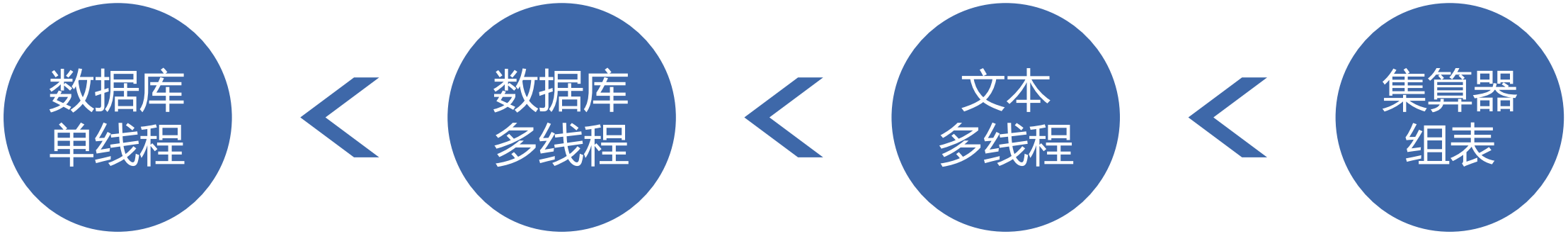
	A	B	C
1	=connect("db")		
2	="select * from 订单 where 订购日期>=date('"/begin/"') and 订购日期<=date('"/end/"')"		
3	select 订单ID,产品ID,单价,数量 from 订单明细		
4	select 客户ID,公司名称 from 客户		
5	select 雇员ID,姓名 from 雇员		
6	select 产品ID,产品名称 from 产品		
7	fork [A2:A6]	=connect("db")	
8		=B7.query@x(A7)	
9	=订单=A7(1)	=明细=A7(2)	
10	=客户=A7(3)	=雇员=A7(4)	=产品=A7(5)
11	>订单.switch(客户ID,客户:客户ID;雇员ID,雇员:雇员ID)		
12	=明细.switch(订单ID,订单:订单ID;产品ID,产品:产品ID)		
13	=A12.new(订单ID.客户ID.公司名称:客户名称,订单ID.订单ID:订单编号,订单ID.雇员ID.姓名:销售,产品ID.产品名称:产品,单价:价格,数量)		

基于文件性能更高

友乾营

如果数据从数据库搬出来，放到文件系统里则可以获得更高的性能

同时，集算器提供了高效的数据存储格式—集文件和组表



取数性能对比

总结

集算器多线程并行的意义在于使用简单、成本低，相对JAVA复杂的多线程编程集算器可以简单到几行脚本，相对数据库集群方案集算器的成本更加可控，而且即使部署数据库集群仍然可以使用集算器加速集群单个数据库节点的取数速度

润乾线上直销系统

好多乾 – 润乾互联网营销

http://www.raqsoft.com.cn/wx/hdq-strategy.html