## 《大数据导论》研读讨论作业

## Main Ideas and Major Contributions of Readings Discussion 3 (in Chinese)

备注：小组讨论翻译并理解给出的研读文献，无需全文直译，但需要撰写文献的核心思想、主要过程和主要贡献，依据撰写质量评价给分

Name: \_\_\_\_\_黄昊\_\_\_\_ Student ID: \_\_20204205\_\_\_\_

In-Memory Big Data Management and Processing: A Survey. IEEE Transactions on Knowledge and Data Engineering, 2015, 27(7):1920-1948.

**核心思想**：本文聚焦于大数据时代下，对已有的内存数据存储系统和处理系统做出了一系列回顾与调查，并介绍了一些框架，并进行了定性比较。在全文的最后，作者总结并提出了内存数据管理的挑战和机遇。

**主要过程**：全文组织架构按如下方式展开，首先介绍内存系统的核心知识，然后分内存数据存储系统和内存数据处理系统，就一些流行的框架分别对其特点做出介绍，然后就不同的框架进行定性比较，最后总结了内存管理系统里面的挑战和机遇。

在内存系统的核心技术部分，作者首先介绍了内存的层次结构：寄存器，缓存，主存和磁盘。存储器到CPU的距离越远，IO效率越低，价格越低，容量越大。然后介绍了内存常见的优化点：寄存器感知优化与缓存感知优化。对于寄存器感知优化而言，传统迭代器的查询处理机制不能利用好数据局部性，而LLVM编译器能实现动态编译，尽可能避免递归函数的调用，从而保持了良好的数据局部性。同时，SIMD能帮助实现数据集并行，显著提高性能。对于缓存感知优化而言，可以重新组织数据布局，优化缓存替换算法，重新组织Cacheline等。

之后，作者介绍了NUMA，其可以提高可以部署在服务器节点中的主内存带宽和总内存大小。经过不断的发展，NUMA的研究方向主要有三个：数据分离，OLTP时延和数据洗牌。在核心技术部分的最后，作者介绍了NVRAM。NVRAM以其非易失性而成为数据库系统中最受欢迎的大容量持久高速存储器。

介绍完核心技术后，作者就内存数据存储系统和内存数据处理系统分别做出相应介绍，并介绍了流行的框架。

就内存数据存储系统而言，作者主要介绍关系数据库和NoSQL数据库，以及缓存系统。

就内存关系型数据库而言，作者介绍了一些常见的框架，其名称及特点如下所列：

* H-store/VoltDB：是分布式的基于行的内存关系数据库，面向高性能OLTP处理。其假设大多数事物模板事先已知，目的是减少运行时事务解析的开销
* Hekaton：面向为高并发OLTP设计，使用无锁或无锁存的数据结构。其采用了编译一次，执行多次的策略：先将SQL语句和存储进程编译为C代码，然后再将C代码转换为机器码执行。采用B-Tree进行索引。
* HyPer/ScyPer：OLTP和OLAP的混合高性能内存数据库，其最大限度地利用了现代硬件特性。其使用动态的查询编译方案，即先将SQL查询编译成汇编代码，然后使用LLVM提供的JIT编译器直接执行汇编代码。有利于寄存器局部性的实现。
* SAP HANA：同时支持面向行和面向列的关系数据存储，提供多种查询语言接口，支持基于时间轴索引的时态查询，提供了基于多版本并发控制的快照隔离。

对于内存NoSQL数据库，其数据结构通常为树、图或KV，而不是表关系。作者在这一部分介绍了以下框架：

* MemepiC：提供作为分布式键值存储的低延迟存储服务，集成了内存中的数据分析功能，支持在线分析
* MongoDB：面向文档的NoSQL数据库，对文档的模式几乎没有限制。其在文档级提供原子性，索引和数据分析只能在单个集合中进行。具有数据存储功能和数据分析功能。
* RAMCloud：分布式内存键值存储，具有低延迟、高可用性和高内存利用率的特点。其采用日志结构的数据组织，内存和磁盘上的数据存储是结构化的。
* Redis：使用键值存储，支持多种数据结构如哈希，列表，集合，有序集合。提供数据持久化机制如快照、仅追加的日志记录。

对于内存缓存系统，其可以为应用程序提供IO优化，以及通过保留结果减轻CPU的负担。其常见的设计目标如下所示：

* 通用型：Memcached，BigTable cache
* 加速分析任务：PACMan和GridGain
* 支持特定框架：net的NCache和用于Windows服务器的Velocity/AppFabric
* 支持严格事务语义的系统：TxCache
* 网络缓存，如HashCache

作者介绍了以下常见框架：

* Memcached：轻量级的键值对象缓存系统，分布式版本是通过客户端库实现的。有针对各种语言的不同版本的客户端库，提供两种主要协议：文本协议和二进制协议，并支持UDP和TCP连接。
* MemC3：促进读密集型工作负载，可以实现高并发性，使用Cuckoo哈希。
* TxCache：基于快照的事务性缓存，用于管理对事务性数据库查询的缓存结果。

以上是内存数据存储系统常见框架，下面介绍内存数据处理系统常见框架。

就内存处理系统而言，有两种类型：专注于批处理的数据分析系统；流处理数据分析系统。下面分别就这两种类型进行介绍：

批处理数据分析系统常见框架如下所列：

* Main Memory MapReduce：支持交互式分析，数据为TB级别且保存在一个节点集群的内存中，具有较高的平均故障时间，但不能保证弹性。输入输出数据缓存在内从中。
* Piccolo：支持跨多节点进行数据分析计算，支持数据位置规范和面向数据的积累。支持自定义累计函数和分区函数，通过一个全局的用户辅助检查点来处理机器故障，通过work stealing机制进行负载均衡。
* SparkRDD：RDD又被成为弹性分布式数据集，可持久化到内存和磁盘，具有轻量级的容错机制，不需要检查点。所有作业以DAG的方式组织，仅当动作中被调用时才使用RDD。

流处理数据分析系统中，举了下列两个框架：

* Spark Streaming：基于Spark开发，面向允许有数秒延迟的应用程序。
* Yahoo!S4：完全分散的分布式流处理引擎，不提供容错机制。

在介绍完常见框架后，作者就流行框架做出了比较，如下表所示：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 框架 | 数据模型 | 工作负载 | 索引 | 并发控制 | 容错机制 | 内存溢出 | 访问处理 |
| 关系型 数据库 | H-Store | 关系型（行） | OLTP | 哈希，B+树，二叉树 | 数据切分， 序列执行，轻量级锁、Spective CC | 命令日志，检查点，副本 | 反缓存 | 存储进程 |
| Hekaton | 关系型（行） | OLTP | 锁存树， 哈希，Bw树 | 乐观MVCC | 日志，检查点，副本 | Project  Siberia | 编译存储 进程 |
| HyPer/ScyPer | 关系型 | OLTP、 OLAP | 哈希， AVL，ART | 虚拟快照，STO， 数据切分，OLTP顺序执行 | 日志，检查点，副本 | 压缩 | JIT、 存储进程 |
| SAP-HANA | 关系型、 图、文本 | OLTP、 OLAP | 时间线索引， CSB+树，变换索引 | MVCC，2PC | 日志，检查点，旁观服务器、GPFS | 表/切分级别 交换，压缩 | 计算图 模型 |
| NoSQL 数据库 | MemepiC | 键值对 | 对象操作、 分析 | 哈希、 跳表 | 虚拟快照、 原子语义特征 | 日志、副本 | 用户空间VMM | JIT |
| MongoDB | 文档 | 对象操作分析 | B树 | 数据库级别锁 | 内存映射文件 | 无 | 无 |
| RAMCloud | 键值对 | 对象操作 | 哈希 | 细粒级锁 | 日志，副本 | 无 | 无 |
| Redis | 键值对 | 对象操作 | 哈希 | 单线程 | 日志、检查点 | 压缩 | 脚本语言 |
| 图 数据库 | Bitsy | 图 | OLTP | 无 | 乐观并 发控制 | 日志、存档 | 无 | 存储进程 |
| Trinity | 图 | 图操作 | 无 | 细粒级 自旋锁 | 副本，TFS | 无 | 存储进程 |
| 缓 存系统 | Memcached | 键值对 | 对象操作 | 哈希 | 细粒级锁 | 无 | 无 | 无 |
| MemC3 | 键值对 | 对象操作 | 哈希 | 乐观锁 | 无 | 无 | 无 |
| TxCache | 键值对 | OLTP | 哈希 | MVCC | 无 | 无 | 无 |
| 大数据 分析系统 | M3R | 键值对 | 分析 | 无 | 切分，锁 | 无 | 无 | 离线 |
| Piccolo | 键值对 | 分析 | 哈希 | 锁 | 检查点 | 无 | 离线 |
| Spark/RDD | 弹性分布式数据集 | 分析 | 无 | 切分， 读写锁 | 世袭，检查点 | 块级交换 | 离线 |
| 实时处 理系统 | Spark Streaming | 弹性分布式数据集 | 流处理 | 无 | 切分， 读写锁 | 世袭，检查点、副本 | 块级交换 | 无 |
| Yahoo!S4 | 事件 | 流处理 | 哈希 | 消息传递 | 旁观服务器 | 无 | 无 |

在介绍完框架后，作者就数据存储系统的机遇与挑战，介绍了以下问题：

* 索引：需要考虑时间效率和空间效率。内存数据库的索引更关注内存和缓存利用率；基于磁盘的数据库索引关注I/O效率。
* 数据布局：缓存相关的设计（如柱状结构、缓存线对齐）和空间利用优化（如压缩、数据碎片整理）是内存数据组织的主要重点。
* 并行：使用多个级别的并行性。
* 并发控制/事务管理：无锁或无锁并发控制机制应用前景广阔，但是需要注意意外中止。
* 查询处理：CPU的高计算能力和编译器使得JIT编译成为可能，可以显著提高查询性能。
* 容错：I/O将会是瓶颈，要尽量减少关键路径上的IO成本，命令日志记录可以减少需要记录的数据，硬件/操作系统辅助的办法前景广阔，如NVRAM和内存映射文件等。
* 数据溢出：解决办法一般分为三个种类：基于用户空间的；基于内核空间的和两者混合的。语义感知有利于在分页中做出更有效的决策，硬件感知和内核方法能提高I/O效率。

最后，作者再次对本文工作进行了总结。其总结了本文的**主要贡献**：

* 介绍了内存数据管理和处理的设计原则，以及设计和实现高性能内存系统的使用技术
* 讨论了一些NewSQL和NoSQL数据库，包括缓存系统、批处理和在线系统。
* 介绍了一些具有发展前景的设计技术和设计原则。