# 大数据技术发展的十个前沿方向（中）

## *Ten Fronties for Big Data Technologies (Part B)*

**吴甘沙**，男，现任英特尔中国研究院院长。2000年加入英特尔，先后在编程系统实验室与嵌入式软件实验室承担了技术与管理职位，期间参与或主持的研究项目有受控运行时、XScale微架构、众核架构、数据并行编程及高生产率嵌入设备驱动程序开发工具等。2011年晋升为首席工程师，共同领导了公司的大数据中长期技术规划，主持大数据方面的研究，工作重点为大数据内存分析与数据货币化。在英特尔工作期间，发表了10余篇学术论文，有23项美国专利（10余项成为国际专利），14项专利进入审核期。

doi: 10.11959/j.issn.2096-0271.2015034

## 6 前沿方向四: 软硬兼施

大数据计算的一个核心要义是软硬两手抓，从具体应用出发，仔细选择软硬件架构实现，并且在运行过程中持续不断地协同优化。

首先，从大数据应用角度出发，硬件架构重点体现在计算、存储、互联。

### 6.1 计算硬件架构

在计算这一块，首先要选择的是大小核。brawny cores是大核，主要是至强服务器芯片，wimpy cores是小核，FAWN（fast array of wimpy nodes）最早展示了小核阵列在特定计算和I/O轮廓下的数据密集型应用中具有一定的能耗优势，其所指的核是Atom或ARM这种级别的核。小核往往需要跟闪存存储结合才能体现系统级的能耗效率提升。对于绝大多数的大数据应用，大核系统优于小核，谷歌公司基础设施的领导者Urs Holzle[21]和亚马逊公司AWS的主要架构师James Hamilton[22]在相关文献中有所论述。另外，基于小核的系统还面临其他一些挑战，比如需要软件做额外的优化、串行部分变慢影响了整体并行加速比、调度无法保证很多小核的高利用率等。

第二，硬件架构方面，要从同构计算到异构计算进行多方面考虑。数据中心成本战略已经从虚拟化（提高IT设备利用率）向每瓦特性能转移，其中，对运行负载做特化（specialization）、走向异构计算是必然趋势。首先是集成异构多核，英特尔公司的E3级服务器芯片在一个芯片上同时集成了CPU和GPU，两者通过LLC（last level cache，三级缓存）进行数据交互，相比基于PCIe加速器或其他高速互联的异构架构更为高效。第二类是独立GPGPU，分立GPU卡做计算加速，比如深度学习框架Caffe就有针对GPGPU的优化。类似地，英特尔公司的Xeon Phi（应用在超级计算机天河2号上，并且帮助获得世界排名第一的计算加速芯片和板卡）将对Caffe做深度的优化。目前来说，GPGPU和Xeon Phi都是插在PCIe插槽上的加速器，它们的加速性能都受到PCIe带宽的限制。2016年Xeon Phi的新版本Knights Landing可以独立启动系统，因此PCIe的限制就不存在了，相对GPGPU方案来说，其可以消除PCIe瓶颈。

另外，FPGA（field-programmable gate array，现场可编程门阵列）加速也是现在广为采用的一种方法，美国自然科学基金会最近的一个报告[6]指出，对于深度学习（比如卷积神经网络），FPGA比GPGPU更有效。百度公司和微软公司都明确声称在生产集群中利用FPGA加速深度神经网络，传言谷歌公司也在做类似尝试。另外，微软公司的Catapult项目还将FPGA应用于Bing搜索引擎网页排名的加速，在功耗增加10%的情况下得到了95%的性能提升[7]。如果英特尔公司对Altera公司的收购顺利完成，这一技术路线将有更多的想象空间。目前来说，FPGA一般用于已训练模型的分类，还不适合对训练过程的加速（比如不容易实现对参数的更新）。

还有一种技术路线是ASIP（application specific instruction processor，专用指令集处理器），最典型的是NPU（neural processing unit，神经处理单元）。总体来说，NPU分两大类：人工神经网络的加速器和类脑芯片（neuromorphic architecture）。前者如中国科学院计算技术研究所DianNao芯片系列[23]和已经商业化的Teradeep（来源于Yann LeCun的工作NeuFlow，一种基于数据流的卷

6 http://insidehpc.com/2014/08/nsf-study-probe-advantages-f/

7 http://research.microsoft.com/pubs/212001/Catapult_ISCA_2014.pdf

积神经网络加速器）；对于类脑芯片，常常见诸媒体的有IBM的TrueNorth芯片和高通的Zeroth芯片，两者都基于更类似于生物神经网络的尖峰神经网络（spike neural network）。TrueNorth展示了在100 mW功耗下模拟复杂的循环神经网络（recurrent neural network）的能力，体现了这一架构的独特之处（但也有人指出功耗优势来自较低的主频）。Zeroth的商业试用并不成功，目前已转向人工神经网络加速器。虽然类脑芯片的商用化还需多年，但它所代表的新型处理范式——计算与记忆的一体化、复杂互联、递归、时空编码、异步、低精度、随机性等特征，有很高的研究价值。目前的努力方向为器件和架构两个方面。美国DARPA的多个项目在架构上做出了有益的探索，包括SyNAPSE（IBM的TrueNorth即来源于此）和Cortical Processor（基于hierarchical temporal memory算法）。在器件方面，DARPA UPSIDE基于模拟芯片，另外还有一些工作基于忆阻器（memristor）。

第三，大数据跟经典高性能计算的区别在于计算和数据的关系。不同于传统意义上的并行计算，大数据以数据密集型应用为主，计算依附于数据，以至于从效率出发，数据必须全部放在内存。由此一个很重要的架构上的变化就是让内存跟处理器能够更加靠近，比如利用eDRAM，使得处理器跟内存之间有更大的缓存；或者把内存和处理器做2.5D/3D堆叠，获得更大的带宽；或者把处理器反过来放到内存里面去，即内存计算（computing in memory）。多年以前学术界曾经尝试主动式存储（active storage），把一些特定的计算置入磁盘控制器，从而更靠近数据。随着闪存和3D XPoint这样的新型非易失内存的普及，内存计算可能得到更多的应用。

### 6.2 存储和互联的优化

大数据存储体系中，大内存加非易失性块存储是性能提升的首选架构。

首先，大内存服务器成为主流。2014年9月，英特尔开发者峰会展示了2U服务器可以容纳1.5 TB内存和100 TB硬盘，使高密度部署更上台阶。而同年发布的Intel E7 v2系列处理器，能够支持高达32-socket的高端服务器，单服务器最高内存容量可达48 TB。这推动了高端大数据一体机的发展。微软在10月份宣布的Azure G系列虚拟机能够提供单虚拟机448 GB内存。与之对应的内存计算需要解决容错的问题，比如Spark的lineage、传统的checkpoint、多数据冗余、基于日志的容错机制等。大内存时代来临以后，这些技术成为了在PC级硬件条件下保障大规模计算的可靠性、大规模计算的成本控制的必不可少的技术基础。

大数据应用同样引导永久性块存储软硬件技术路线产生了重大改变。从SSD（solid state drives，固态硬盘）到PCIe SSD到闪存存储，这一路的发展经历了硬件局部变革、软件进步、软硬件共同演进的3个阶段历程。

硬件局部变革：SSD是以硬件为主的进步，传统的硬盘文件系统架构基本保留。

软件进步：PCIe SSD开始触及块设备管理和文件系统软件栈，各种轻量级、多任务优化的文件系统软件栈蓬勃发展。传统文件系统针对毫秒级访问时延优化，而闪存将时延缩短到微秒级，并且支持1 000 000/s读写操作以上的并发度，势必需要新的文件系统设计。

软硬件共同演进：全闪存存储开始出现，同时文件系统发生巨大变革甚至被放弃，许多新型应用系统可以让应用直接

操作闪存里面的数据块，实现相当程度的DRAM和flash的统一寻址，最大化性能提升。未来的NVRAM（nonvolatile random access memory，非易失随机存取存储器）[24]，能够兼具数据永久性和数据动态访问性能，所以数据静态保存和动态使用的状态彻底统一，文件系统需求降低，序列化和反序列化重要程度下降，整个系统栈会更浅、更简单，性能更优异。在这些NVRAM中，英特尔公司的3D XPoint最受瞩目，其性能和耐用性都将达到目前NAND闪存的1 000倍，接近于DRAM，这将进一步模糊内存和存储的界限。该技术预期2016年上市，可以用在几乎所有存储用途——主内存、缓存或块存储。如果该技术性价比按照摩尔定律降低，势必引导大数据服务器设计发生重大变革。

大数据中常见的聚合（aggregation）、置换（shuffle）等操作对互联带宽和通信时延提出了很高的要求。物理层面，如今节点和节点之间的网络带宽已经达到40 Gbit/s，未来基于全光交换结构的设计能达到数百Gbit/s甚至Tbit/s，比如英特尔公司的硅光（silicon photonics）技术。分立的高性能NIC将逐渐消失，处理器（板）与其他节点处理器（板）的内存直接通信。软硬件界面上，为了发挥内存计算的效用，需要更便宜、更高扩展性的RDMA互联解决方案。纯软件部分，一些新的网络协议被应用于数据中心，比如data center TCP（DCTCP）针对Fork-Join结果的网络流量进行了优化。另外，Spark采用了类似BitTorrent的协议来减少shuffle产生的节点间网络流量。

### 6.3 软硬件协同优化

随着硬件的巨大进步，软件架构逐渐成为瓶颈，软硬件架构协同的优化成为必然。数据中心出现了重新垂直化（re-verticalization）的趋势。各家大厂都针对硬件特点优化软件栈，而做法各有不同。

（1）把硬件特性暴露给软件栈

相比于过去逐步抽象隔离的多层次架构设计，新发展方向强调核心信息上下沟通，这体现了螺旋上升的技术发展模式。一个值得注意的趋势是一些互联网和大数据巨头都建立了自有编译器团队，甚至是自有Java虚拟机团队。流行的Hadoop、Spark等开源分布式系统都运行在Java虚拟机上面。经典的Java虚拟机具有对软件隔离硬件细节的特性，实现了软件架构的平台无关的巨大优势，但同时这个特性也大大削弱了软件针对硬件优化的潜力。当大数据从以推广为主过渡到以性能应用为主的阶段后，就需要通过一种方式来打破这个界限。比如英特尔公司尝试的NativeTask[25]，把MapReduce中间结果的重排过程从Java代码转移到原生代码，并且针对系统缓存架构进行优化，MapReduce的总体系统效能提升了30%~50%。Spark最近创造了Terasort的世界纪录，其中网络通信也通过Netty利用了原生代码优化库。现在很多基于Java的大数据计算系统，都选择利用底层特殊优化的线性代数原生库来提升性能。

即使不采用原生代码，仍需要对软件进行内存访问模式的优化，尽量使随机访问发生在CPU缓存，内存访问量尽可能降低，而对块设备的操作基本是顺序读写。另外，Java虚拟机带来的另一个开销是内存管理，在大内存模式下垃圾回收器（garbage collector）效率降低，管理开销可能占总体运行时间的10%以上，应对方案除了虚拟机采用针对大内存高度优化的并行、并发回收算法，还出现了硬件加速的垃圾回收[8]。另外，应用本身也必须采用如Tachyon这样的并发存储管理的内存文

8 http://link.springer.com/article/10.1007%2Fs11227-010-0493-0

件系统，多个应用自主把共享的数据放入Tachyon，从而有效减少单个应用内存的堆需求，降低管理开销。

最新的例子是Spark从2014年开始的Tungsten项目[9]。Databricks公司在项目中为性能目标引入3项重大改进：自主内存管理，以降低Java对象管理和垃圾回收机制的开销；CPU Cache-aware Computation，从数据结构和算法上针对处理器缓存作优化；直接代码生成，更好地利用现代编译器的硬件优化功能。

（2）重新设计软件栈

如前文所述的全闪存存储和NVRAM，或者从更高角度，以硬盘为代表的块设备也正在IP化、对象存储化，传统的块设备软件栈正在重新设计。

（3）多服务器范围或者机架范围的一体机设计

现在服务器商业模式不再是简单地销售服务器硬件或者单纯销售软件，而是把软件、硬件打包成一体机，推出一个以应用为导向的整体解决方案。最有名的一体机就是SAP的HANA。一体机方案是从某个具体应用出发，以机架为单位，对服务器单机配置、块设备、网络互联、单机操作系统、分布式管理系统、分布式应用等进行大深度和大广度的优化定制。比如HANA的B+树是针对高速缓存优化的，单机上它能用SSE对数据进行流式、在线的压缩和解压缩，多机上能够高效地执行任务分配和结果汇总。IBM的Netezza一体机则从高性能在线数据分析出发，采用了基于FPGA的软硬件协同设计。

（4）从传统数据中心大数据机群建设转向私有云/公有云为基础的大数据服务

这代表一种行业发展方向。以Hadoop基础架构为例，在早期各个公司普遍致力于建设自有数据中心内的大规模Hadoop机群，这里面有早期Hadoop套件对多租户支持差的原因。随着云平台商业模式的发展和Hadoop套件的进化，各种以“Hadoop as Servive”为主打卖点的云端大数据基础设施服务成为行业新热点，极大地降低了新兴公司利用大数据技术的门槛。而后来者主推Spark的Databricks公司从一开始就认识到云服务模式，Databricks Cloud是该公司的主要运作模式之一。

大数据云化发展中，除了商业模式创新，还包括大数据平台技术和云端虚拟平台技术的深度整合。因为从一般意义上来说，大数据平台如Hadoop、HDFS和虚拟技术是性能不兼容的，其中涉及虚拟化本地数据I/O接口的效率问题。各家公司都为此做出了不懈努力，如VMWare公司开发了Hadoop virtual extensions[10]技术，专门解决Hadoop软件栈的效率；Hadoop和Spark社区都和Docker社区紧密合作，提高协同效率。

第二个大方向是硬件可重构。最初的硬件可重构是从块存储设备开始逐步发展的。传统硬盘机柜只能在SAN、NAS、DAS 3种模式静态选择一种固定配置，由于传统高性能计算和大数据计算本质不同，两种平台不能运行时共享单一硬件。SeaMicro开发的freedom fabric storage技术允许动态划分硬盘组，从而解决这个问题，极大地提高了硬件利用率。

工业界继续推广数据中心硬件重构的概念。英特尔公司推出了Rack Scale Architecture，鼓励机群设计人员把眼光从服务器单机放宽到全机架，从3个突破口出发，根据应用对硬件进行横向整合。第一个突破口是按类别构建资源池。传统服务器设计单机内部集成计算、内存、存储和网络各个模块。重构以后单机只保留最基础的启动系统的硬件，尽可能把大内存、大容量块设备、大型加速器等单独组成模块，并且为机架内所有服务器所用。第二个

9 https://databricks.com/blog/2015/04/28/project-tungsten-bringing-spark-closer-to-bare-metal.html

10 http://www.vmware.com/files/pdf/techpaper/Virtualized-Hadoop-Performance-with-VMware-vSphere6.pdf

突破口是提升高速互联的速度和层级，互联带宽从万兆提高到100 Gbit/s以上，并且革新协议栈，使RDMA成为标准配置。从单机互联，发展到各个资源模块通过新型技术直接互联。第三个突破口是用软件对资源池动态划分、动态挂载到各个应用的虚拟服务器，从而实现大数据应用的核心需求，在机架尺度上针对应用负载进行动态优化。

在当下的大数据应用中，软硬件协同优化的最成功、最普及的范例是基于神经网络的深度学习系统。各个业内著名互联网公司纷纷构建了大规模的机群，专用于针对视觉和语音的深度学习。而后在系统运营过程中纷纷进行软硬件协同优化提升学习系统的效率。英特尔公司资助参与了GraphLab和Petuum的开源系统。

## 7 前沿方向五：多、快、好、省

“多、快、好、省”就是在实现数据量（多）和计算性能（快）的前提下，易于编程和管理（好），追求资本和运维成本最低（省）。

### 7.1 垂直化优化

实现“多、快、好、省”的方式，首先是垂直打通技术栈、做跨层次的优化。第6节举例说明了基于闪存块设备的软硬件协同设计，这里以内存计算为例描述一个新方向。它不是对技术栈的单独某一层技术进行改造或者革新，而是从下到上的“大手术”。

在硬件平台层，主张配置大容量内存（DRAM），搭配大容量SSD甚至全闪存，未来采用3D XPoint内存，将大容量非易失存储器的内存直接可寻址化，板级之间用RDMA互联，支持NVM express over fabrics。

在系统软件的数据管理和存储层，主流方案是内存数据库、内存文件系统和具有内存缓存的分布式文件系统（如Tachyon、HDFS），特点是优化的内存缓存、基于堆外大内存的Java虚拟机优化、基于内存文件系统的进程间文件数据交换。

在系统软件的计算框架层面，强调应用全新的分布式内存计算平台。所谓的内存计算，术语是in memory computing，初始主要指那些优化数据存储方式，在内存（DRAM）中完成数据存储、检索、分析的数据库技术，最新的发展方向被称为内存中的数据网格（in memory data grid）技术。Spark进一步发展了这个概念，除了在内存中完成SQL/HQL计算以外，Spark能够综合利用分布式机群的所有内存来完成复杂机器学习计算。

在应用层面，尤其是数据可视化层面，重新设计数据结构、注重数据模型创新，是实现“多、快、好、省”的主要措施。在数据实时查询和可视化领域，一个代表性的例子是NanoCube。通过独创的基于树的数据存储和查询算法，对于上百万甚至上亿的数据集，NanoCube可以在普通笔记本电脑的内存中完成索引，空间利用率很高。在数据分析领域，一个鲜明的例子是图计算框架GraphChi。GraphChi的目标是在单机上实现大数据计算，其技术基础为：新的分层的图计算架构，支持流式图计算；支持实时更新的图计算。

### 7.2 降低空间和时间的复杂度

#### 7.2.1 降低空间复杂度

降低空间复杂度最直观的做法就是把大数据变小。变小的方式有很多，具体

如下。

（1）对数据压缩几乎已经是所有大数据架构的标配，除了Hadoop采用的几种压缩算法，2013年初谷歌公司开源了Zopfli，在最大压缩率下的数据比zlib小3%~8%。压缩率与消耗的CPU时间相关，Zopfli消耗的CPU时间比zlib大2~3个数量级，所以更适合变化性较小的数据（压缩一次即可）。多数系统的压缩解压缩发生在内存和I/O之间，也就是说内存里已经是解压的数据。也有少数系统，如IBM的Netezza，数据在内存中仍保持压缩状态，FPGA配合CPU实时地边解压边处理，进一步提升了内存空间效率。

（2）列式存储，例如Apache Parquet，能够使压缩率获得数量级的提升，而它本身也是舍大得小的办法，因为往往查询并不需要访问所有列，所以数据加载时无须读入不相干的列。当然，好的SQL优化器也对行存储进行修剪。

（3）多维度存储的缓存机制，找出最重要的大数据并放在内存和闪存里面，以加速计算主体。

（4）采用稀疏的结构，比如在进行商品推荐应用时，用户和购买商品的映射矩阵是核心数据结构，通过采用各种稀疏存储结构就能够把映射矩阵的空间复杂度下降。

（5）在大数据规模下，处理所产生的中间数据是比较困难的问题。比如MapReduce和Spark在shuffle操作中如果粒度不当，可能产生大量的中间数据。除了采用合适的粒度外，Spark采取了一种新的shuffle机制，从基于散列的分桶变成了基于sort的分桶，极大地提高了内存使用效率。无论是Terasort创造世界纪录，还是在实际应用场景，如大规模的广告分析逻辑回归（logistics regression），Spark的新机制都扮演了重要角色。

### 7.2.2 降低时间复杂度

另外一方面是时间复杂度的降低。

（1）尝试“简单”、高可变（high variance）的模型。这些模型虽然计算上较为简单，但特征较为复杂，在小数据场景下，往往会过拟合。但更多的数据解决了这一问题，使得模型的功效急剧增长。比如谷歌公司Peter Norvig的文章[26]指出，数据赋予了简单模型不可名状的魔力，以Web文本处理为例，$n$-gram相当简单，但这类模型具有较大的特征空间（词汇表里每一个单词作为一个特征），经过大量的数据训练，机器翻译有了一个质的提升。同时，相比较而言，简单模型的计算复杂度较低，计算效率提升很显著。

（2）使用简单模型的组合。模型组合（ensemble）是一个机器学习的专业领域。例如，Netflix曾经通过竞赛，百万美元奖励推荐有效性提升10%的创新。很多团队参加这个竞赛，但是没有单个团队能达到10%；后来若干个团队之间进行模型的组合，一举超越10%。在美国版开心辞典Jeopardy上打败人类冠军的IBM Watson系统，采用了一百多种模型的组合。当数据增大时，ensemble公司往往也能获得更好的边际效益。值得一提的是模型组合要考虑计算复杂性问题，Netflix公司就是因为获奖者组合模型计算量太大而放弃了它的实际应用。但一般而言，假设每种模型是$O(N^2)$复杂度，当$N$非常大时，多个模型的组合远远比一个$O(N^3)$的复杂模型更高效。

（3）采样和近似。采样是统计工具中重要的数据处理方法，在大数据里面它被赋予更多含义。大数据处理中，因为数据体量的问题，查询处理基本只能以批处理方式进行，实时响应在理论上是不可能的。工业界大胆引入采样技术以应对实时要求，依靠精心设计的针对具体应用场景

优化的原始数据采样算法，可以实现TB级别数据规模下秒级时延的不精确结果的查询响应，比如BlinkDB。近似则用在另外一个领域。如果能够确保应用对输入精度不敏感，就可以在数据处理中有意识地放宽条件、引入误差，从而通过降低复杂度或提升并行性来大幅度降低处理时间。经典的广告应用场景有Bloom filter、CountMinSketch算法等。

（4）降维和混合建模。考虑到在多数应用中大数据有两个特点经常同时出现：高纬度和稀疏数据点，所以如果能共同应用各种稀疏数据结构的优化，必将能够极大地提升计算效率。

所有这些方法都能降低机器学习的复杂度，需要从应用出发灵活运用。

### 7.3 分布式和并行化

大数据中分布式优化的经典案例是从ACID到BASE（basically available, soft state,eventually consistent）[11]的变化。ACID是传统关系数据库事务特性的必然条件，但是其代价高昂，成为数据库规模和性能双提升的致命瓶颈。BASE只选中其最终一致性，结果NoSQL数据库的规模和性能实现双赢，后者实现了并行的可能。

对于迭代计算，计算数学上有许多例子，经典的是线性方程求解的两种常用近似方法，雅可比（Jacobi）迭代法和高斯—赛德尔（Gauss-Seidel）迭代法。对于Jacobi方法，当前迭代必须基于上一迭代的结果数据，这就给分布式优化带来巨大困难，因为每个迭代的结果数据分布在许多节点上，细粒度的计算同步和数据同步抵消了分布式优化的收益。对于Gauss-Seidel方法，当前迭代可以不依赖最新数据，结果同样收敛，只是理论上的收敛速度会下降，但是辅以分布式优化，整体的计算时间反而大大节省。因此，很多新开发的机器学习算法能够容忍模糊性，能够基于本地节点上过时的数据（不一定是其他节点发送过来的最新数据）进行计算，打破迭代之间数据的依赖。这样使所有节点能够尽可能快地并行计算，但同时算法依然能够收敛。深度学习领域在这个方面迅猛发展，谷歌公司采用了参数服务器（parameter server）的方式，CMU既有参数服务器，也有更通用的SSP（stale synchronous parallel）[27]。CMU的Petuum进一步发展了SSP的思想，针对超大模型的机器学习实现数据并行、模型并行和调度层面的任务并行。

并行化是分布式的必须手段。分布式机器学习依赖于一种并行方法或者几种并行方法的组合。首先是数据并行，然后是图并行或者模型并行。模型并行需要基于模型或图结构进行数据划分、任务调度，使用比较广泛的是GraphLab，较为新颖的是Petuum。

总体来说，并行化和分布式的核心诉求就是减少通信、减少同步、提高效率。从分布式系统实践出发，一定会碰到许多问题，诸如缓存问题、一致性问题、本地化问题、划分问题、调度问题、同步粒度问题等。这些问题有多种解决方式，需要针对具体问题实际考虑。比如同步问题就有BSP全同步、GraphLab异步或者SSP的半同步等多种选择；状态更新可以批量进行，也可以个别进行；数据通信可以单次传输全部数据，也可以只传输改变的数据。

总结起来，“多、快、好、省”有赖于软硬件的协同优化、时空复杂度的降低以及分布化和并行化。

## 8 前沿方向六：天下三分

在传统计算时代，系统研究者常常

11 http://queue.acm.org/detail.cfm?id=1394128

希望能够设计出一套软硬件架构覆盖尽可能多的问题领域。2000年后，Michael Stonebraker提出要针对不同的计算需求提供不同的计算引擎，效果会更好（database is not a one-size-fits-all category），他因为在这一方向上的长期实践获得了2015年的图灵奖，由此出现了数据和计算的分野。

（1）数据类型

大数据时代的主流特征是提升优化了很多原来传统高性能计算不太关注的数据类型。首先，最被关注的数据结构是表格——键值对（key-value）。基于关系型数据库的表格是结构化数据，利于并行。各种长度未知的相关性未知的键值对序列，就欠缺结构化。大数据NoSQL数据库专门优化键值对查询。其次，图结构也被重视，比如图计算，大数据大大提升了图计算的重要性，把它提升到一种编程语言、分布式处理框架的高度来解决问题。再如矩阵和数组，大数据大大拓展了对不规则矩阵和不规则多纬数组的支持，比如SciDB。

（2）计算范式

大数据的兴起是源于计算范式的突破。

首先是MapReduce编程模型的创新、推广和改进。MR是一种批处理计算，主要出发点有：数据规模太大，把计算任务发送到数据节点；简化任务间通信模型，去掉底层显示的同步编程，用抽象的数学范式来重构整个并行计算，降低学习门槛，简化分布式支撑平台设计。

其次是以图结构为基础的计算范式。

（3）编程模型

大数据平台的成熟激发了大数据编程模型的繁荣，目前主流的编程模型有：数据并行、流式计算的任务并行、图并行。

尤其值得一提的是，大数据为图计算带来的机会是图并行，因为图结构的稀疏性，多数顶点只有较小的邻域，这样，在一个迭代中整个图可以划分为很多并行处理的邻域。图并行中的计算依赖必须有一致性保证，如GraphLab的可变一致性模型。

目前的最新发展方向又要求在图结构上支持关系操作，比如加入一张表，Spark的GraphX最早实现了这一能力，接着一些图计算编程模型（如GraphLab）也加入了这种特性。还有一种概率编程模型[12]，适用于一些基于概率图的机器学习，它的未来趋势有待观察。

大数据领域的另外一个特殊领域是reactive programming或事件驱动编程模型，主要面向对大规模并发事件有实时响应要求的场景，比如海量广告点击、海量金融交易等。一个非常流行的范式叫reactive范式。最早在Erlang里，叫actor模式。如今在Scala Akka语言里面优化了异步处理逻辑，有更好的可扩展性。基于这一模型，英特尔中国开发了一个新的开源项目GearPump，目前获得广泛好评[13]。

总体而言，针对应用特性选择特殊定制方案是当前大数据实现3V的主要思路。

12 http://probabilistic-programming.org/wiki/Home

13 http://downloads.typesafe.com/website/presentations/GearPump_Final.pdf

## 9 前沿方向七：分久必合

大数据爆发以来，各种新编程语言编程方式层出不穷、百家争鸣。经过一段时间积淀，各种编程语言之间发生了融合。

Big Dawg[14]是英特尔公司在MIT最近支持的研究工作，Stonebraker也参与了这一工作。Big Dawg提出了普适的编程模型，一种叫做BQL的语言，它支持关系和线性代数、复杂数据模型、迭代计算、并行计算。这种语言能够把计算映射到底层的不同计算和存储引擎。

Twitter Summingbird是在编程接口层面融合，同时支持批量（MapReduce，

14 http://istc-bigdata.org/index.php/building-a-new-application-to-hardware-management-stack-for-big-data/

Spark）和流式（Storm）的计算。

Lambda架构则是在应用框架层面的融合，尝试用一个编程框架把实时流计算和批量计算统一，以高效率的批处理为基础，辅以流处理实现增量计算，综合起来几乎形成一个完美的大数据框架[15]。Nathan Marz声称这一模型可以突破CAP理论的限制。

Spark则是在实现框架层面的融合，如图1所示。

微软的REEF则是通过资源管理层（如YARN）来实现多计算模型的融合。YARN在大数据不同计算范型的融合上起到了至关重要的作用。

Databricks公司以Spark为核心，雄心勃勃地开发BDAS软件栈，目的是实现一个多层面、多模型融合的面向多种应用的统一大数据处理平台。基础性的底层构件是HDFS，它负责永久性存储，在HDFS之上搭建Tachyon内存文件系统，以提升访问效率。在计算框架上，以Spark内存计算框架为核心，重大改进是引入Velox的模型管理，这可以看作数据并行和模型并行的一种尝试。框架之上，就是各种计算范式的支持模块，比如流计算有Spark streaming的流计算、关系数据库，NonSQL应用有SparkSQL，图计算有GraphX，机器学习有MLlib等。

除了Databricks，Cloudera也不甘人后，Hadoop创始人Doug Cutting在以“The One Platform Initiative”为副标题的网络演讲中提出引入Spark作为Hadoop下一个通用处理框架，以补充MapReduce。更宏观地讲，就是促使BDAS和Hadoop的融合。

传统的大规模并行编程（MPP）和Hadoop/MapReduce各有擅长，现在也在融合。早期的混合架构，用MPP处理大规模的结构化数据，用Hadoop/MR处理海量的半结构化、非结构化数据。最新的趋势是两者开始融合，高性能计算/超算跟大数据开始朝向统一平台发展，出现了高性能数据分析（high performance data analysis）和数据密集型计算（data intensive supercomputing）等新的子领域。英特尔公司已经开始推动“one single system for HPC and big data”。

在基础设施层面，hyper-converged infrastructure更是势不可挡。

天下三分之后，分久必合。这个合与分并不矛盾，基于负载特性在应用层划分，再在总体架构层面汲取共性、打通彼此来“合”，是符合事物的发展规律的。

15 http://lam bda-architecture.net/

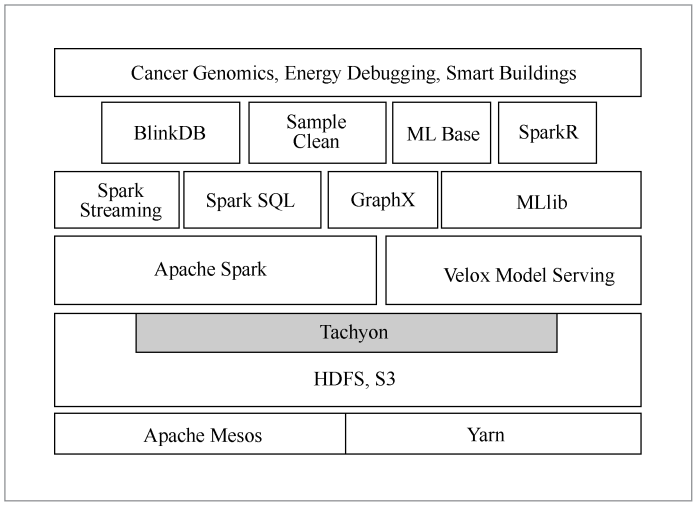


图1 Spark 架构

## 参考文献

[21] Holzle U. Brawny cores still beat wimpy cores, most of the time. IEEE Micro, 2010, 30(4)

[22] Hamilton J. Cooperative expendable micro-slice servers (CEMS): low cost, low power servers for Internet scale services. http://mvdirona.com/jrh/TalksAndPapers/JamesHamilton_CEMS.

pdf, 2009

[23] Chen T S, Du Z D, Sun N H, *et al*. Diannao: a small-footprint high-throughput accelerator for ubiquitous machine-learning. ACM Sigplan Notices, 2014, 49(4): 269~284

[24] Mittal S, Vetter J S, Li D J. A survey of architectural approaches for managing embedded DRAM and non-volatile on-chip caches. IEEE Transactions on Parallel and Distributed Systems, 2014, 26(6): 1524~1537

[25] Dong Y, Yin X S, Lian C, *et al*. Unleash the architecture power to accelerate big data processing. Journal of Computer Science and Technology, 2014

[26] Halevy A, Norvig P, Pereira F. The unreasonable effectiveness of data. IEEE Intelligent Systems, 2009, 24(2): 8~12

[27] Ho Q R, Cipar J, Cui H, *et al*. More effective distributed ml via a stale synchronous parallel parameter server. Advances in Neural Information Processing Systems, 2013: 1223~1231 □