《计算机系统设计》课程报告

湖南大學

HUNAN UNIVERSITY

选题名称：各测试系统架构分析

姓　　名：袁 纯 楸

学　　号：201708010225

专业班级：智能 1701

信息科学与工程学院

一、 Linpack 标准测试程序及其分析

Linpack 测试包括三类，Linpack100、Linpack1000 和 HPL。

Linpack100：求解规模为 100 阶的稠密线性代数方程组，它只允许采用编译优化选项进行优化，不得更改代码，甚至代码中的注释也不得修改；

Linpack1000：求解规模为 1000 阶的线性代数方程组，达到指定的精度要求，可以在不改变计算量的前提下做算法和代码上做优化。

HPL（高度并行计算基准测试）：对数组大小 N 没有限制，求解问题的规模可以改变，除基本算法（计算量）不可改变外，可以采用其它任何优化方法。

LINPACK 压力测试的目的主要为检测系统中 CPU 的工作的稳定性及内存访问的稳定性。

安装过程比较复杂，略：

相关配置文件：

在 HPL 测试中，使用的参数选择与测试的结果有很大的关系。HPL 中参数的设定是通过从一个配置文件 HPL.dat 中读取的，所以在测试前要改写 HPL.dat 文件，设置需要使用的各种参数，然后再开始运行测试程序。配置文件内容的结构如下：

```
HPLinpack benchmark input file //文件头
Innovative Computing Laboratory, University of Tennessee//说明性文字

HPL.out output file name (if any)//如果使用文件保留输出结果，设定文件名
6 device out (6=stdout,7=stderr,file)//输出方式选择（stdout,stderr 或文件）
“device out”=“6”时，测试结果输出至标准输出（stdout）
“device out”=“7”时，测试结果输出至标准错误输出（stderr）
“device out”为其它值时，测试结果输出至第三行所指定的文件中

2 # of problems sizes (N) //指出要计算的矩阵规格数量
```

1960 2048Ns //每种规格分别的数值

• 矩阵的规模 N 越大，有效计算所占的比例也越大，系统浮点处理性能也就越高；但矩阵规模 N 的增加会导致内存消耗量的增加——当系统实际内存空间不足，使用缓存、性能会大幅度降低。

• 计算规模的大小，应以设置的大页面内存总量做计算，计算方式为：N*N*8=大页内存总量*0.8，内存总量换算为字节，规模的大小最好为 384 的倍数。

2 # of NBs //指出使用不同的分块大小数量

60 80 NBs //分别指出每种大小的具体值

提高数据的局部性，从而提高整体性能，HPL 采用分块矩阵的算法。

分块的大小对性能有很大的影响，NB 的选择和软硬件许多因素密切相关 NB 值的选择主要是通过实际测试得到最优值；NB 大小的选择还跟通信方式、矩阵规模、网络、处理器速度等有关系。

1 PMAP process mapping (0=Row-,1=Column-major)

选择处理器阵列是按列的排列方式还是按行的排列方式。

按列的排列方式适用于节点数较多、每个节点内 CPU 数较少的系统；

按行的排列方式适用于节点数较少、每个节点内 CPU 数较多的大规模系统。

2 # of process grids (P x Q) //指出用进程组合方式数量

2 4 Ps //每对 PQ 具体的值

2 1 Qs

二维处理器网格（P×Q）的要求：

P×Q = 系统 CPU 数 = 进程数。（一般：一个进程对于一个 CPU 可以得到最佳性能；Intel Xeon：关闭超线程可以提高 HPL 性能）

• P≤Q：P 的值尽量取得小一点，列向通信量（通信次数和数据量）>>横向通信。

• P=2n：L 分解的列向通信采用二元交换法，此时性能最优。

• 在集群测试中，P×Q = 系统 CPU 总核数。

16.0 threshold //余数的阈值

说明测试的精度。做完线性方程组的求解以后，检测求解结果是否正确：

若误差在这个值以内就是正确，否则错误。

若是求解错误，其误差非常大；若正确，则很小。

1 # of panel fact //用分解方法数量

1 PFACTs (0=left, 1=Crout, 2=Right) //具体那种分解方法

0 left,1 crout,2 right

1 # of recursive stopping criterium //停止递归的判断标准

4 NBMINs (>= 1) //具体的标准数值（须不小于 1）

1 # of panels in recursion //递归中用分割法的数量

2 NDIVs //一种 NDIV 值为 2，即每次递归分成两块

1 # of recursive panel fact. //用递归分解方法的数量

2 RFACTs (0=left, 1=Crout, 2=Right) //这里每种都用到（左，右，crout 分解）

L 分解的方式：

在消元过程中，HPL 采用每次完成 NB 列的消元（L 分解），然后更新后面的矩阵。

对每一个小矩阵作消元时，都有 3 种算法：L、R、C，分别代表 Left、Right 和 Crout。

1 # of broadcast //用广播方法的数量

3 BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM) //指定具体哪种（有 1-ring,1-ring Modified,2-ring,2ring Modified,Long 以及 long-Modified）

L 的横向广播方式：

HPL 中提供了 6 种广播方式：前 4 种适合于快速网络；后 2 种采用将数据切割后传送的方式，主要适合于速度较慢的网络。目前，机群系统一般采用千兆以太网甚至光纤等高速网络。在小规模系统中，选择 0 或 1；对于大规模系统，选择 3

1 # of lookahead depth //用几种向前看的步数

1 DEPTHs (>=0) //具体步数值（须大于等于 0）

横向通信的通信深度：依赖于机器的配置和问题规模的大小。

2 SWAP (0=bin-exch,1=long,2=mix)//交换算法（bin-exchange，long 或二者混合）

64 swapping threshold //采用混合的交换算法时使用的阈值

U 的广播算法：列向广播

HPL 提供了 3 种 U 的广播算法：二元交换法、Long 法和二者混合法。

SWAP="0"，采用二元交换法；SWAP="1"，采用 Long 法；SWAP="2"，采用混合法。

0 L1 in (0=transposed,1=no-transposed) form //L1 是否用转置形式

0 U in (0=transposed,1=no-transposed) form //U 是否用转置形式表示

L 和 U 的数据存放格式：若选择“transposed”->采用按列存放，否按行存放。

1 Equilibration (0=no,1=yes) //是否采用平衡状态

在回代中使用，一般使用其默认值

8 memory alignment in double (> 0) //指出程序运行时内存分配中的采用的对齐方式

为内存地址对齐设置：用于在内存分配中对齐地址。出于安全考虑-选择 8。

为调试出高的性能，必须考虑内存大小，网络类型以及拓扑结构，调试上面的参数，直到得出最高性能。

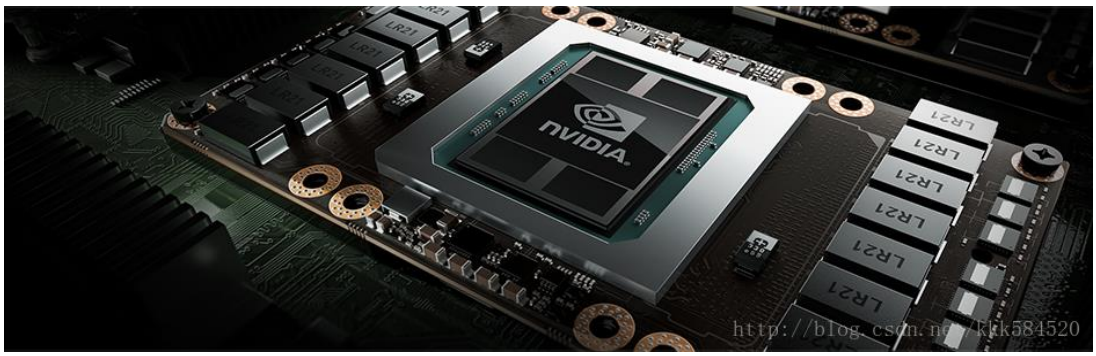
二、 Tesla GPU 架构分析

以 Tesla P00 为例：

其 NVIDIA 官网相关参数：

	适用于基于 PCIe 的服务器的 P100	适用于 NVLink 优化服务器的 P100
双精度浮点运算能力	4.7 teraFLOPS	5.3 teraFLOPS
单精度浮点运算能力	9.3 teraFLOPS	10.6 teraFLOPS
半精度浮点运算能力	18.7 teraFLOPS	21.2 teraFLOPS
NVIDIA NVLink 互联带宽	-	160 GB/s
PCIe x16 互联带宽	32 GB/s	32 GB/s
CoWoS HBM2 堆叠式显存容量	16 GB or 12 GB	16 GB
CoWoS HBM2 堆叠式显存带宽	732 GB/s or 549 GB/s	732 GB/s
提升使用页面迁移引擎编程的能力	✓	✓
ECC 保护助力实现可靠性	✓	✓
针对数据中心部署优化服务器	✓	✓

Tesla P100 采用顶级大核心 GP100，面积 610 mm^2，如图中间位置：



GP100 参数汇总如下：

芯片：GP100

架构：SM_60

工艺：16 nm FinFET

支持：双精度 FP64，单精度 FP32，半精度 FP16

功耗：250 W

CUDA 核心数：3584（56 SMs，64 SPs/SM）

GPU 时钟（Base/Boost）：1189 MHz/1328 MHz

PCIe：Gen 3 x16

显存容量：12/16 GB HBM2

显存位宽：3072/4096 bits

显存时钟：715 MHz

显存带宽：539/732 GB/s

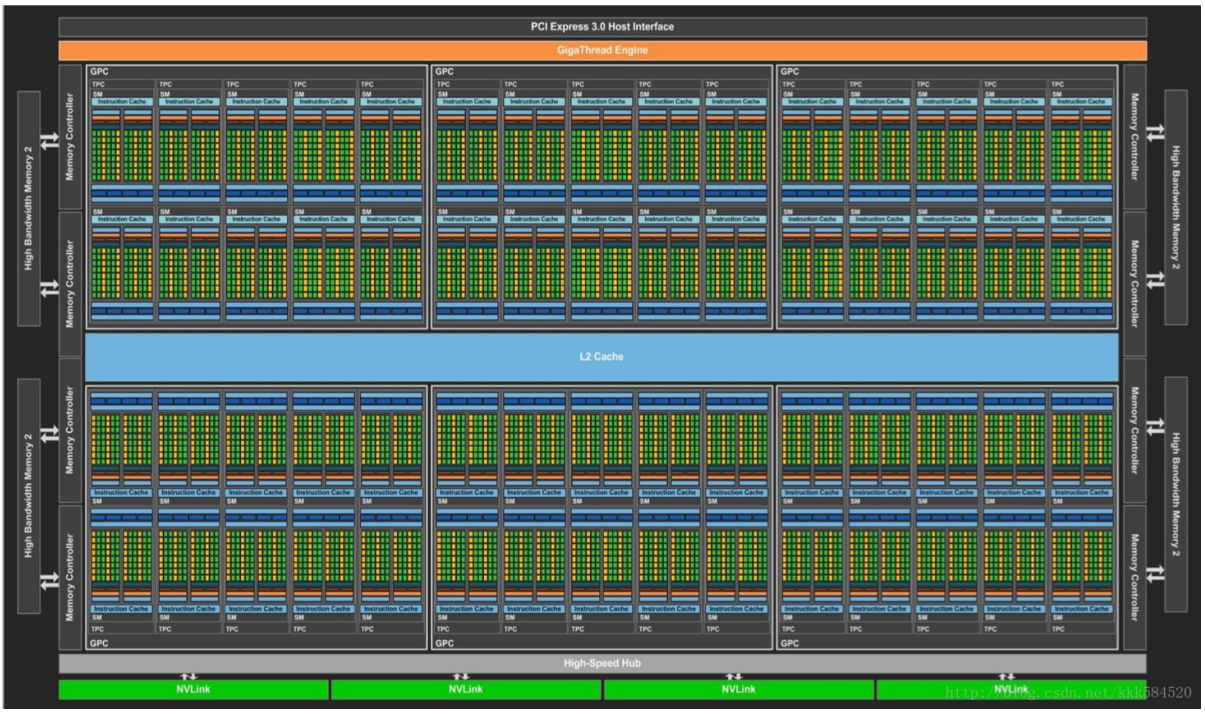
DGX-1 拥有 8 颗帕斯卡架构 GP100 核心的 Tesla P100 GPU，一共组成了 128 GB 的显存。

浮点运算（FP16）达到了 170 TFlops（官方宣称相当于 250 个 x86 服务器），内置 7 TB 的 SSD，两颗 16 核心的 Xeon E5-2698v3 以及 512 GB 的 DDR4 内存，整机功耗 3200W，售价为 129000$。

其 GP100 GPU 硬件架构：

GP100 称：最高性能并行计算处理器。

GP100 包含一组 GPC（图形处理簇，Graphics Processing Clusters）、TPC（纹理处理簇，Texture Processing Clusters）、SM（流多处理器，Stream Multiprocessors）以及内存控制器。其架构如下：



一颗完整的 GP100 芯片包括 6 个图形处理簇，60 个 Pascal 流多处理器，30 个纹理处理簇和 8 个 512 位内存控制器（总共 4096 位）。

每个图形处理簇内部包括 10 个 流多处理器。

每个流多处理器内部包括 64 个 CUDA 核心和 4 个纹理单元。

Tesla P100 只用了 GP100 上 60 个 SM 中的 56 个。

GP100 的第六代 SM 架构提高了 CUDA 核心利用率和能效，核心频率更高，整体 GPU 性能有较大提升。

GP100 的 SM 包括 64 个单精度 CUDA 核心。而 Maxwell 和 Kepler 的 SM 分别有 128 和 192 个单精度 CUDA 核心。虽然 GP100 SM 只有 Maxwell SM 中 CUDA 核心数的一半，但总的 SM 数目增加了，每个 SM 保持与上一代相同的寄存器组，则总的寄存器数目增加了。这意味着 GP100 上的线程可以使用更多寄存器，也意味着 GP100 相比旧的架构支持更多线程、warp 和线程块数目。与此同时，GP100 总共享内存量也随 SM 数目增加而增加了，带宽显著提升不至两倍。

一个 SM 的架构：

其中绿色的“Core” 为单精度 CUDA 核心，共有 64 个，同时支持 32 位单精度浮点计算和 16 位半精度浮点计算，其中 16 位计算吞吐是 32 位计算吞吐的两倍。图中橘黄色的 “DP Unit” 为双精度计算单元，支持 64 位双精度浮点计算，数量为 32 个。每个 GP100 SM 双精度计算吞吐为单精度的一半。

下面处理性能表证实：

PERFORMANCE SPECIFICATION FOR NVIDIA TESLA P100 ACCELERATORS

	P100 for PCIe-Based Servers	P100 for NVLink-Optimized Servers
Double-Precision Performance	4.7 TeraFLOPS	5.3 TeraFLOPS
Single-Precision Performance	9.3 TeraFLOPS	10.6 TeraFLOPS
Half-Precision Performance	18.7 TeraFLOPS	21.2 TeraFLOPS

Pascal SM 架构图：一个 GP100 SM 分成两个处理块，每块有【 32768 个 32 位寄存器 + 32 个单精度 CUDA 核心 + 16 个双精度 CUDA 核心 + 8 个特殊功能单元(SFU) + 8 个存取单元 + 一个指令缓冲区 + 一个 warp 调度器 + 两个分发单元】

- Pascal SM 架构相比 Kepler 架构简化了数据通路，占用面积更小，功耗更低。
- Pascal SM 架构提供更高级的调度和重叠载入/存储指令来提高浮点利用率。
- GP100 中新的 SM 调度器架构相比 Maxwell 更智能，具备高性能、低功耗特性。
- warp 调度器（每个处理块共享 1 个）在一个时钟周期内分发两个 warp 指令。

双精度算法是很多 HPC 应用（如线性代数，数值模拟，量子化学等）的核心。GP100 的一个关键设计目标就是显著提升这些案例的性能。

GP100 中每个 SM 都有 32 个双精度（FP64）CUDA 核心，即单精度（FP32）CUDA 核心数目的一半。GP100 和以前架构相同，支持 IEEE 754-2008 标准，支持 FMA 运算，支持异常值处理。

注意：在 Kepler GK110 中单精度核心数目：双精度核心数目为 3:1。

使用 FP16 计算相比 FP32 可以带来 2 倍性能提升，数据传输时间也可以大大降低。

注意：GP100 上，两个 FP16 运算可以使用一个双路指令完成。

相比上一代产品和上上代产品情况如下表所示：

Table 1. Tesla P100 Compared to Prior Generation Tesla products

Tesla Products	Tesla K40	Tesla M40	Tesla P100
GPU	GK110 (Kepler)	GM200 (Maxwell)	GP100 (Pascal)
SMs	15	24	56
TPCs	15	24	28
FP32 CUDA Cores / SM	192	128	64
FP32 CUDA Cores / GPU	2880	3072	3584
FP64 CUDA Cores / SM	64	4	32
FP64 CUDA Cores / GPU	960	96	1792
Base Clock	745 MHz	948 MHz	1328 MHz
GPU Boost Clock	810/875 MHz	1114 MHz	1480 MHz
Peak FP32 GFLOPs[1]	5040	6840	10600
Peak FP64 GFLOPs[1]	1680	210	5300
Texture Units	240	192	224
Memory Interface	384-bit GDDR5	384-bit GDDR5	4096-bit HBM2
Memory Size	Up to 12 GB	Up to 24 GB	16 GB
L2 Cache Size	1536 KB	3072 KB	4096 KB
Register File Size / SM	256 KB	256 KB	256 KB
Register File Size / GPU	3840 KB	6144 KB	14336 KB
TDP	235 Watts	250 Watts	300 Watts
Transistors	7.1 billion	8 billion	15.3 billion
GPU Die Size	551 mm^2	601 mm^2	610 mm^2
Manufacturing Process	28-nm	28-nm	16-nm FinFET

[1] The GFLOPS in this chart are based on GPU Boost Clocks.

表现出了优异的高性能和高能效。

Maxwell 架构相对 Kepler 做了改进，提升了效率。

Pascal 基于 Maxwell 架构，利用 TSMC 16 nm FinFET 工艺进一步降低了能耗。

计算能力：

GP100 GPU 支持最新 6.0 计算能力。

Table 2. Compute Capabilities: GK110 vs GM200 vs GP100

GPU	Kepler GK110	Maxwell GM200	Pascal GP100
Compute Capability	3.5	5.2	6.0
Threads / Warp	32	32	32
Max Warps / Multiprocessor	64	64	64
Max Threads / Multiprocessor	2048	2048	2048
Max Thread Blocks / Multiprocessor	16	32	32
Max 32-bit Registers / SM	65536	65536	65536
Max Registers / Block	65536	32768	65536
Max Registers / Thread	255	255	255
Max Thread Block Size	1024	1024	1024
Shared Memory Size / SM	16 KB/32 KB/48 KB	96 KB	64 KB

三、 Infiniband 网络结构分析

InfiniBand 技术不是用于一般网络连接的，它的主要设计目的是针对服务器端的连接问题的。因此，InfiniBand 技术将会被应用于服务器与服务器（比如复制，分布式工作等），服务器和存储设备（比如 SAN 和直接存储附件）以及服务器和网络之间（比如 LAN， WANs 和 the Internet）的通信。

1、 IB 基本概念：

IB 是以通道为基础的双向、串行式传输，在连接拓朴中是采用交换、切换式结构(Switched Fabric)，在线路不够长时可用 IBA 中继器(Repeater)进行延伸。每一个 IBA 网络称为子网(Subnet)，每个子网内最高可有 65,536 个节点(Node)，IBA Switch、IBARepeater 仅适用于 Subnet 范畴，若要通跨多个 IBASubnet 就需要用到 IBA 路由器(Router)或 IBA 网关器(Gateway)。

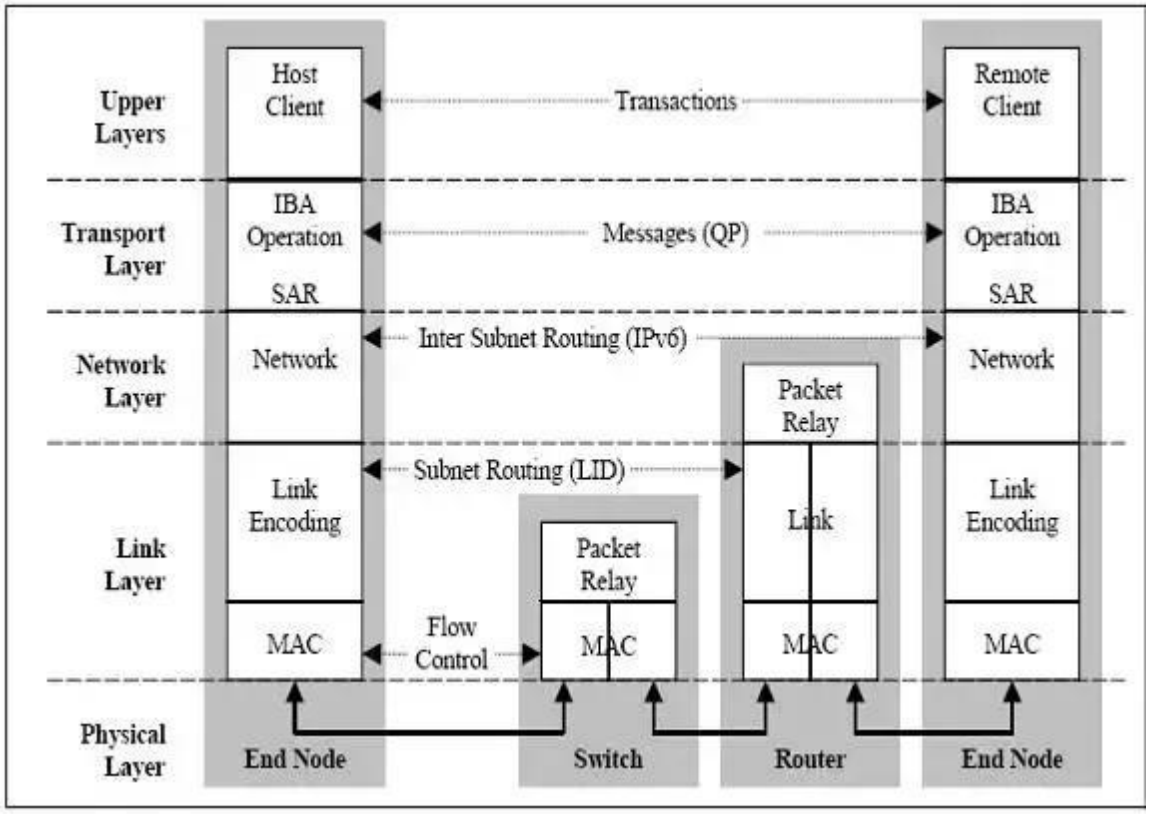
每个节点(Node) 必须透过配接器(Adapter)与 IBA Subnet 连接，节点 CPU、内存要透过 HCA(Host Channel Adapter)连接到子网；节点硬盘、I/O 则要透过 TCA(TargetChannel Adapter)连接到子网，这样的一个拓扑结构就构成了一个完整的 IBA。

IB 的传输方式和介质相当灵活，在设备机内可用印刷电路板的铜质线箔传递(Backplane 背板)，在机外可用铜质缆线或支持更远光纤介质。若用铜箔、铜缆最

远可至 17m，而光纤则可至 10km，同时 IBA 也支持热插拔，及具有自动侦测、自我调适的 Active Cable 活化智能性连接机制。

2、 IB 协议简介：

InfiniBand 也是一种分层协议(类似 TCP/IP 协议)，每层负责不同的功能，下层为上层服务，不同层次相互独立。 IB 采用 IPv6 的报头格式。其数据包报头包括本地路由标识符 LRH，全局路由标示符 GRH，基本传输标识符 BTH 等。



A、 物理层

物理层定义了电气特性和机械特性，包括光纤和铜媒介的电缆和插座、底板连接器、热交换特性等。定义了背板、电缆、光缆三种物理端口。

并定义了用于形成帧的符号(包的开始和结束)、数据符号(DataSymbols)、和数据包直接的填充(Idles)。详细说明了构建有效包的信令协议，如码元编码、成帧标志排列、开始和结束定界符间的无效或非数据符号、非奇偶性错误、同步方法

等。

B、 链路层

链路层描述了数据包的格式和数据包操作的协议，如流量控制和子网内数据包的路由。链路层有链路管理数据包和数据包两种类型的数据包。

C、 网络层

网络层是子网间转发数据包的协议，类似于 IP 网络中的网络层。实现子网间的数据路由，数据在子网内传输时不需网络层的参与。

数据包中包含全局路由头 GRH，用于子网间数据包路由转发。全局路由头部指明了使用 IPv6 地址格式的全局标识符(GID)的源端口和目的端口，路由器基于 GRH 进行数据包转发。GRH 采用 IPv6 报头格式。GID 由每个子网唯一的子网 标示符和端口 GUID 捆绑而成。

D、 传输层

传输层负责报文的分发、通道多路复用、基本传输服务和处理报文分段的发送、接收和重组。传输层的功能是将数据包传送到各个指定的队列(QP)中，并指示队列如何处理该数据包。当消息的数据路径负载大于路径的最大传输单元(MTU)时，传输层负责将消息分割成多个数据包。

接收端的队列负责将数据重组到指定的数据缓冲区中。除了原始数据报外，所有的数据包都包含 BTH，BTH 指定目的队列并指明操作类型、数据包序列号和分区信息。

E、 上层协议

InfiniBand 为不同类型的用户提供了不同的上层协议，并为某些管理功能定义了消息和协议。InfiniBand 主要支持 SDP、SRP、iSER、RDS、IPoIB 和 uDAPL 等上层协议。

• SDP(SocketsDirect Protocol)是 InfiniBand Trade Association (IBTA)制定的基于 infiniband 的一种协议，它允许用户已有的使用 TCP/IP 协议的程序运行在高速的 infiniband 之上。

• SRP(SCSIRDMA Protocol)是 InfiniBand 中的一种通信协议，在 InfiniBand 中将 SCSI 命令进行打包，允许 SCSI 命令通过 RDMA(远程直接内存访问)在不同的系统之间进行通信，实现存储设备共享和 RDMA 通信服务。

• iSER(iSCSIRDMA Protocol)类似于 SRP(SCSI RDMA protocol)协议，是 IB SAN 的一种协议 ，其主要作用是把 iSCSI 协议的命令和数据通过 RDMA 的方式跑到例如 Infiniband 这种网络上，作为 iSCSI RDMA 的存储协议 iSER 已被 IETF 所标准化。

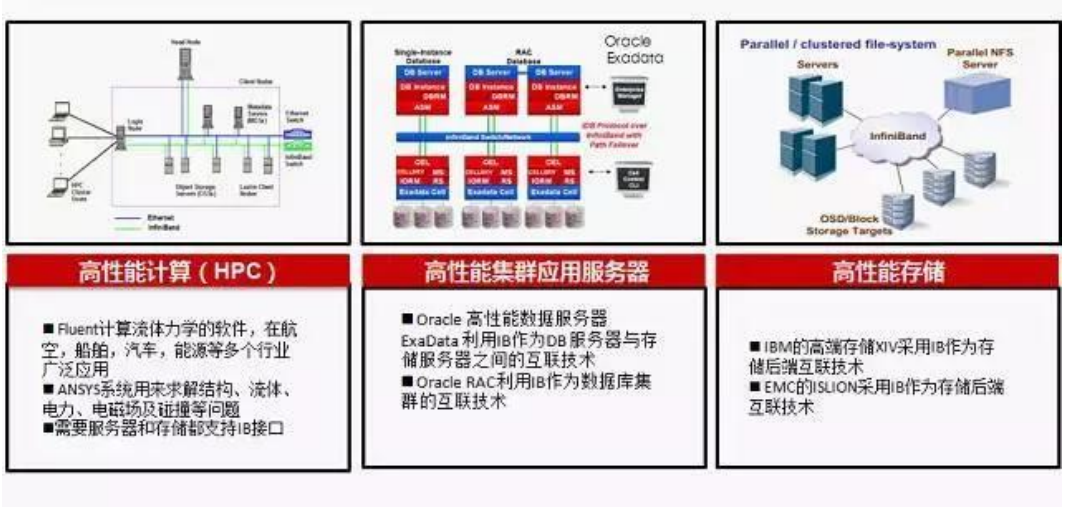
• RDS(ReliableDatagram Sockets)协议与 UDP 类似，设计用于在 Infiniband 上使用套接字来发送和接收数据。实际是由 Oracle 公司研发的运行在 infiniband 之上，直接基于 IPC 的协议。

•IPoIB(IP-over-IB)是为了实现 INFINIBAND 网络与 TCP/IP 网络兼容而制定的协议，基于 TCP/IP 协议，对于用户应用程序是透明的，并且可以提供更大的带宽，也就是原先使用 TCP/IP 协议栈的应用不需要任何修改就能使用 IPoIB。

• uDAPL(UserDirect Access Programming Library)用户直接访问编程库是标准的 API，通过远程直接内存访问 RDMA 功能的互连（如 InfiniBand）来提高数据中心应用程序数据消息传送性能、伸缩性和可靠性。

3、 IB 应用场景

Infiniband 灵活支持直连及交换机多种组网方式，主要用于 HPC 高性能计算场景，大型数据中心高性能存储等场景，HPC 应用的共同诉求是低时延(<10 微秒)、低 CPU 占有率（<10%）和高带宽(主流 56 或 100Gbps)



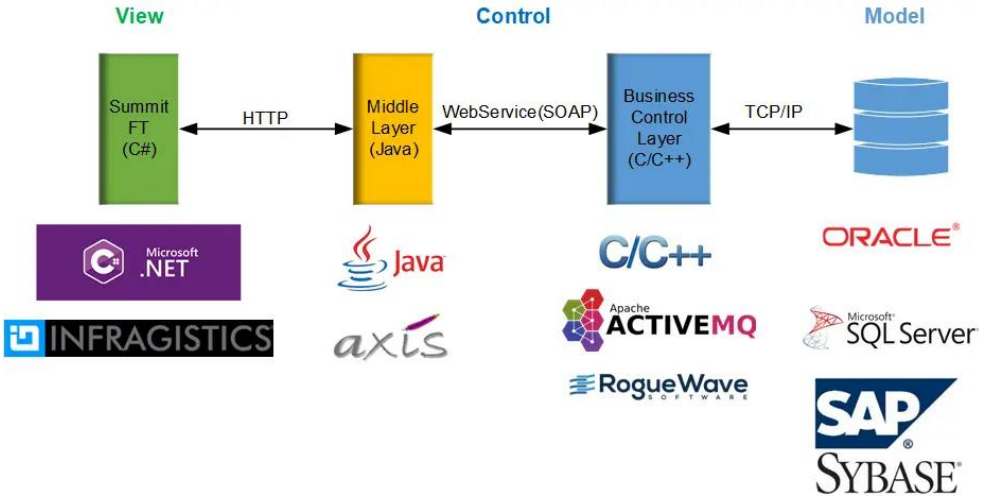
• Infiniband 在主机侧采用 RDMA 技术释放 CPU 负载，可以把主机内数据处理的时延从几十微秒降低到 1 微秒；

• InfiniBand 网络的高带宽(40G、56G 和 100G)、低时延(几百纳秒)和无丢包特性吸取了 FC 网络的可靠性和以太网的灵活扩展能力。

四、 Summit 架构分析

Summit 系统是由 Misys(现已被 Finastra 收购)公司开发的一套用于银行、券商进行银行间(主要)资金交易管理的系统。

集前台交易录入；中台额度管理、合规管理、估值管理、工作流管理；后台清算管理、账务处理于一身的前中后台一体化系统。Summit 尤其擅长后台，后台功能非常强大。开发时间大概在 1999 年左右，直到目前，仍然占据全球 60%的主要银行客户。



Summit 系统是典型的 MVC 结构系统，其中：

A、 View 层：称为 SummitFT，基于微软 C# .NET 技术；

SummitFT 使用 Infragistics 的 C#控件库作为基础，封装出了一套独属控件。整个界面风格统一、控件布局合理，操作方便，对用户比较友好。作为对比，Calypso 基于 Java 做的界面；Kondor 基于 C 做的界面，操作体验上来说，跟 SummitFT 是没法比。

B、 Control 层:分为 2 部分,一部分为 Java 开发的通信中间层,另一部分为 C/C++编写的 Summit 主体部分；

Summit 作为典型的 CS 程序，客户端与服务端通讯采用的不是 TCP/IP 直接通讯的方式，而是采用了 HTTP 协议和 WebService 的方式。

其中，SummitFT 通过 HTTP 协议与通讯中件层通讯；通讯中间层采用 WebService 与 etoolkit 进程通讯，达到使用 Summit 后端服务的目的。

优点：Control 层不仅可以对接 SummitFT，还提供了一套灵活的供其他客户端调用的方式，比如 Summit 就支持 VBA、Java 等其他语言的直接调用。由此可以看出，Summit 系统在设计时已经考虑到了系统的开放性。通讯中间层采用 Java 语言编写，负责接收 SummitFT 的 HTTP 连接，并负责 HTTP 协议报文与 SOAP 报文之间的转换。Summit Business Control 层即上文提到的 etoolkit，etoolkit 使用 C/C++开发，实际上就是一个 WebService Server，负责处理中间层的请求，并将结果封闭成 SOAP 报文，返回给通讯中间层。

C、 Model 层：Summit 业务数据抽象、存取层，基于 ENTITY 实现，支持主流的 Oracle/SQL Server 以及 Sybase 数据库。

Model 层依赖 Summit 数据抽象 ENTITY 以及关系型数据库，目前支持 Oracle, Sybase 以及 SQL Server。Model 层进行 Summit 数据的序列化与反序列化。ENTITY 即 Summit 系统的元数据，在 Summit 系统中，所有的数据（交易数据、静态数据、系统基础数据）都以 ENTITY 进行抽象。ENTITY 不仅包含属性(Properties)，还会包含接口(Interface)和具体的方法(Method)。因此，ENTITY 完全可以用现在的面

向对象来理解。