A Study of HPC

李海峰

December 27, 2022

本文旨在对 HPC 的原理、应用、技术框架、发展趋势等进行一个基础的了解，以便掌握 HPC 基本的背景知识并为 HPC 在我司的应用提供参考。

Contents

1 HPC 简介

High-performance computing (HPC) 是指使用超级计算机或者计算机集群来解决需要大量计算的复杂计算问题，通常需要集成多个处理器构成一个计算机系统，并且还需要协同进行分布式计算和并行计算。[36]

借助于 HPC，可以将原本需要计算数年的任务缩短到几天甚至更短，对于 HPC 而言，最关键的指标就是其算力究竟有多强。衡量算力通常使用每秒浮点运算次数 *(floating point operations per second)*[34] 来表示, 其单位为 $FLOP/s$。在具体计算时，又可以分为两种方式:

- **计算理论最大性能 (peak)**: 通过计算时钟频率乘以单个时钟周期完成双精度浮点数操作次数得出，如 2.5 $Ghz \times 4\ FLOP/clock = 10\ GFLOP/s$，但是受限于 I/O 操作等，实际不可能达到
- **计算真实性能 (real)**: 通过一些程序进行持续计算并得出观测结果，例如Top500使用Linpack进行线性代数运算。

2019 年超算就可以执行超过千万亿次 FLOPS。相比之下，高端游戏台式机的速度要慢 1,000,000 倍以上，可执行约 200 千兆次 FLOPS (1 × 109)[1]。

1.1 HPC 的应用领域

HPC 最早是应用在数学和物理学领域，如今 HPC 的应用范围已经非常广泛，主要围绕在一些超算问题解决上面 [29]:

- **偏微分方程求解**: 包括物理、工业上的一些方程求解，如麦克斯韦方程、Black-Scholes 方程等，以及流体力学研究、天气预报等

- **线性代数**: 包括搜索引擎 PageRank、有限元模拟、线性代数建模等
- **两两耦合相互作用的大型系统**: 包括宇宙学、分子动力学模拟等
- **图问题**: 包括机器学习、欺诈识别、数据分析等
- **随机系统**: 包括粒子物理学研究、核反应堆设计、模拟疾病传播等

抛开具体的领域，单从问题本身对于算力、存储等的要求来看，HPC 可以解决的问题可以归纳为以下几类 [9]:

- **计算密集型**: 解决问题需要进行大量的计算工作, 譬如进行金融建模、风险敞口等，这类问题是传统 HPC 应用最多的类别
- **内存密集型**: 解决问题需要使用大量的内存, 相对于计算密集型问题，这类问题对内存的大小、不同节点内存的性能差异以及 CPU、内存的交互上更敏感
- **数据密集型**: 解决问题时需要处理大的数据集，即“大数据”问题
- **高吞吐量型**: 需要批量计算大量的无关任务，例如大量的任务运行，彼此间无需 CPU 级别通信，在单独的线程上并行运行。例如，一家企业可能向某节点集群中的各个处理器核心提交了 1 亿条信用卡记录 [47]。

除了传统的超算问题外，一些新兴的领域也在探索与 HPC 进行融合，成为 HPC 的重要应用场景，包括:

- **高性能数据分析 (HPDA)**: 为了支持日益复杂的算法,高性能数据分析（HPDA）已成为一个将 HPC 资源应用于大数据的新细分领域。包含统计分析、数据清洗、数据汇聚等，借助 HPDA 可以将大数据处理系统（如 Hadoop、Spark）等与 HPC 融合起来, 一方面实现全流程的加速处理，另一方面大数据也可以与 HPC 的技术相互受益 [17, 46]
- **人工智能 (AI/ML/DL)**: 包含三大类应用，一类是以卷积网络为核心的图像检测、视频检索类问题，如安防、自动驾驶等；一类是以强化学习为核心的博弈决策类问题，如交通规划；一类是以 Transformer 为核心的自然语言处理类问题，如搜索推荐、智能人机接口等 [46]
- **高级建模和模拟**: 高性能计算建模和模拟可用于药物研制和测试，汽车和航空航天设计、气候/天气系统预测以及能源应用 [1]。

一份研究预测 HPDA 的市场增长从 2018 年到 2023 年为 15.4%，超过全球 HPC 整体市场的预测增长 7.8%，而 HPC AI 预测增长为 29.5%，远超 HPDA[20]。

1.2 HPC 发展历史

HPC 的发展大致经过了以下几个阶段 [14]:

- **1940s-1960s**: 诞生首个超级计算机。二次世界大战以及冷战催生了超级计算机的出现，来完成密码破译，以及核武器、飞机、潜艇设计等
- **1975-1990**: Seymour Cray（被誉为超级计算机之父）成立公司并发展了一系列超级计算机，如 Cray-1 ~4
- **1990-2010**: 由单机转为集群以便能够利用更多的内存和 CPU
- **2000-至今**: 进入混合加速的集群时代，由多核 CPU、GPU 加速节点等组成混合集群，对特定计算进行优化

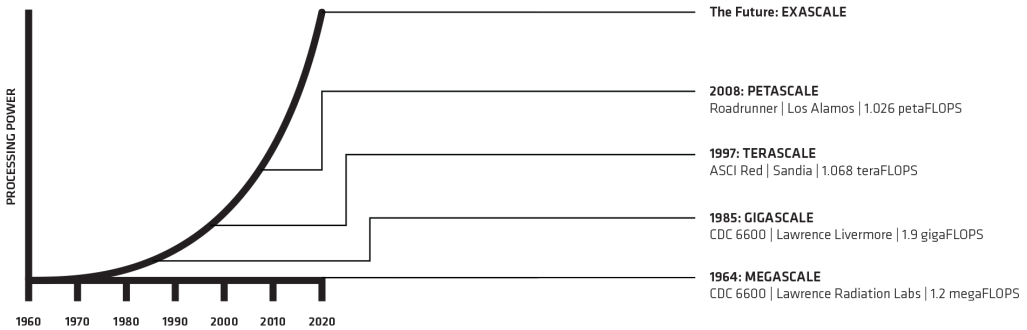


Figure 1: 世界 HPC 算力的变革

截至 2022 年 9 月，Top500 上排名第一的超级计算机为 Frontier（美国），其计算速度已达到 1.102 $ExaFLOP/s$。

我国在过去十年依托顶尖的超算系统，大规模并行应用设计和研制方面取得显著进步，多个超算应用入围戈登贝尔奖（ACM Gordon Bell Prize）[1] 包括：

[1] 戈登贝尔奖是国际高性能计算应用领域最高奖, 设立于 1987 年, 主要颁发给高性能应用领域最杰出成就。

年份	应用
2014	地震模拟
2016	大气动力框架、相场模拟、海浪模拟
2017	地震模拟、大气模拟
2018	图计算框架
2021	量子模拟、人造太阳、第一性原理

1.3 HPC 发展趋势

传统的 HPC 计算以超算为代表，超级计算机代表着高性能计算系统最尖端的水平。由各个超算中心或者实验室开发，尽管我国的超算能力处于世界领先水平，但也存在着一系列的问题 [46]:

- 国产超算平台架构多样，应用移植和调优工作量大，例如神威·太湖之光（10,649,600 核，异构众核处理器 SW26010，OpenACC*，Athread + MPI）与天河 2 号（498 万核，2xCPU+2xMatrix2000，OpenMP,OpenCL+MPI）架构就不一样，程序在不同的超算系统间进行移植十分困难，而国外超算架构逐渐收缩到加速器异构，几乎是 Intel CPU + NVIDIA GPU
- 对支持复杂应用全流程计算的能力不足
- 受制于美国定向打击制裁

随着云计算的成熟，云平台强大的计算资源也成为了 HPC 的新基建。除了超算中心外，民用和商用 HPC 也有着较大的发展，包括微软、戴尔、Intel、IBM、AWS 等均在 HPC 占有较大的市场份额。国内有并行科技、阿里云、华为云、腾讯云等均有开放 HPC 业务。

近年来 HPC 发展有如下的趋势:

- **异构高性能计算**: 将 CPU 和 GPU 等不同类型的计算单元结合在一起，被称为异构计算。常见的计算异构单元类别包括 CPU、GPU、ASIC、FPGA 等。
- **向云上迁移**: HPC 正朝向云端发展，一方面是因为问题复杂度越来越超出本

地 HPC 的算力；一方面，云计算允许按需提供资源，在成本和弹性上更有优势（公共云）；一方面，在 HPC 中采用容器技术的势头也愈发显著 [44, 47]。调查显式，2019 年已有超过 10% 的 HPC 任务运行在云上，除了公有云外，私有云、混合云也在快速发展 [20]。

- **与大数据融合**: HPC 有望与大数据融合，即通过同一大规模计算机集群来分析大数据，运行模拟和其他 HPC 负载 [47, 46]。
- **与 AI 融合**: HPC 与 AI 有着较为类似的架构要求，AI 应用在 HPC 上可以在同等精度上更快地获取结果，将两者融合是一个新的应用趋势 [15, 12, 46, 45]。

2 传统 HPC 理论和应用基础

2.1 并行计算与加速比

普通计算机执行任务计算时，大多需要按照处理流程将任务划分为多个子任务，子任务需要等待其他依赖任务执行完毕，而 HPC 使用更多的资源、更多的算力来一次执行更多的任务，解决问题的时间依赖于可用的资源、以及算法的设计。也就是说，通过更好的硬件、更多的资源、以及更好的算法如并行计算等来提高系统的算力。HPC 中有两种数据处理模式 [1]:

- **串行处理**: 由 CPU 完成，每个 CPU 核心通常每次只能处理一个任务
- **并行处理**: 可利用多个 CPU 或 GPU 完成，GPU 最初是专为图形处理而设计的。它可在数据矩阵（如屏幕像素）中同时执行多种算术运算。同时在多个数据平面上工作的能力使 GPU 非常适合在机器学习 (ML) 应用任务中进行并行处理，如识别视频中的物体。

根据阿姆达尔定律 (Amdahl's law)，固定负载的计算任务在并行化之后的效率提升，可以用以下公式来表示 [33]:

$$S_{\text{latency}}(s) = \frac{1}{(1-p) + \frac{p}{s}}$$

其中，

- S_{latency} 为理论加速比
- s 为处理器数量
- p 为可以并行处理的部分比例

阿姆达尔定律下，增加并行处理器的数目最终会达到一个极限的加速比，受限于程序设计中的串行部分，如 fig. 2所示。

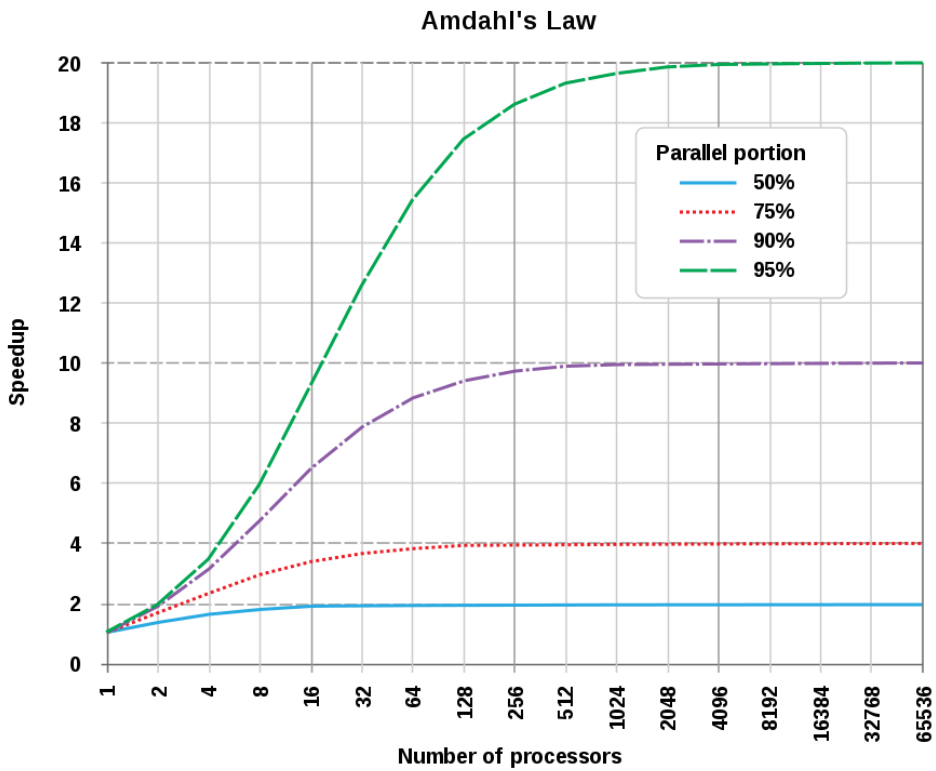


Figure 2: 增大处理器数目并不能无限增大加速比，这取决于程序的可并行度

阿姆达尔定律假设问题的大小是固定的，串行执行的总耗时是固定的，但在实际问题中，一个重要的事实是，并行计算不仅可以进行加速，也可以为解决问题提供更大的空间，可以处理更大规模的问题，而其又可以拆分为更多独立的小问题来并行解决 [31]。因此，另一个更优的模型是古斯塔夫森定律 (Gustafson's law)[35]，它可以表示为：

$$S = s + p \times N \tag{1}$$
$$= s + (1 - s) \times N \tag{2}$$
$$= N + (1 - N) \times s \tag{3}$$

其中，

- S 为理论加速比
- N 为处理器数量
- s、p 分别为程序花费在串行和并行计算上部分所占的比例，满足 $s + p = 1$

根据这一定律，可以得出一个更符合实际应用的加速比与处理器关系图，如 fig. 3所示。

2.2 HPC 系统的计算模型

HPC 系统需要使用高性能的硬件，包括 CPU、加速器、以及存储和网络设备等；除此之外，还需要一个顶层的设计，来决定计算负载将如何被执行。在进行并行计算时，有两种模型，分别是共享内存模型和分布式内存模型两种 [19]，参见 fig. 5。它们的特点如 fig. 4所示。

HPC 系统有以下几种设计模式 [16]:

- **并行计算**: 按照共享内存模型，使用成百上千的处理器进行协同计算
- **集群计算**: 使用相同的计算机组成集群，使用高速的本地网络进行互连，并使用中心化的任务调度和资源管理来执行计算任务。整个集群组成了一个 HPC 系统。
- **网格计算**: 通过网络将多个计算机连接起来，它们可以同处一个本地网络或者分布于不同的地理位置。相对于集群计算而言，网格计算可以使用不同类型的计算机，网格中的计算机可以向集群贡献剩余算力。不同网格中的计算机使用低速网络互连，但每个网格可以实现自治。

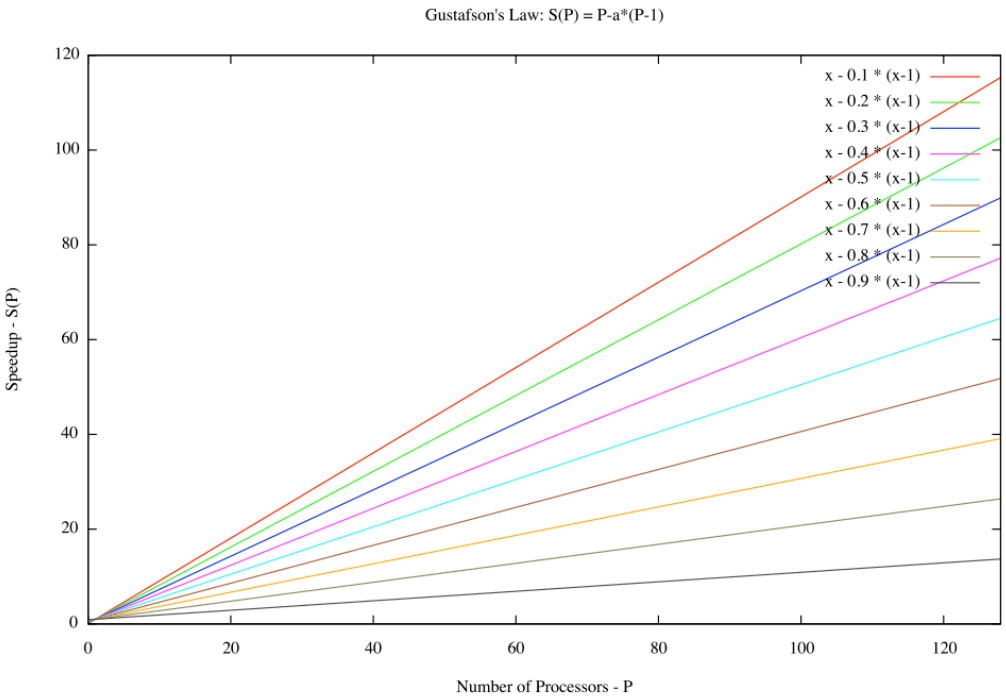


Figure 3: 根据古斯塔夫森定律得出的计算比与处理器的关系

2.3 硬件设施及网络拓扑

典型的 HPC 集群架构如 fig. 6所示。其中，SMS（System Management Server）为控制节点，通过 BMC[2] 网络与计算节点联通。

2.3.1 高速网络

除开以太网外，HPC 集群中还可以增加高速网络来传输，例如进行消息传递、并行文件系统传输等，主要包含 InfiniBand、Omni-Path[3]等。

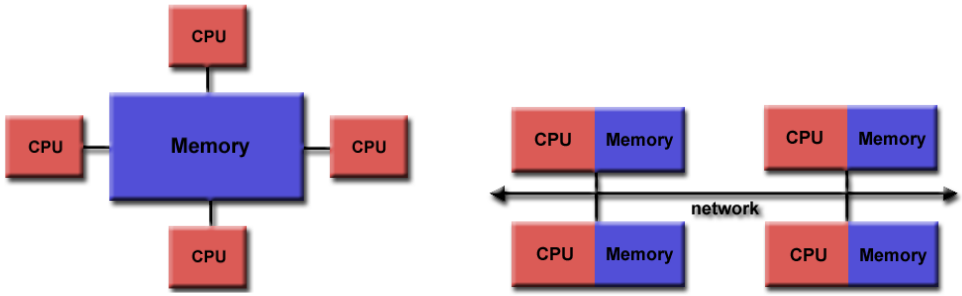
InfiniBand(“无限带宽”) 是一个用于高性能计算的计算机网络通信标准，它具有极高的吞吐量和极低的延迟，用于计算机与计算机之间的数据互连。InfiniBand

[2]BMC: 基板管理控制器 (Baseboard Management Controller), 它可以在机器未开机状态下对机器进行固件升级、查看机器设备等操作。

[3]Omni-Path 由 Intel 推出，但不就宣布放弃此项技术 [39]。

共享内存模型	分布式内存模型
• 所有处理器都可以在同一地址空间访问所有内存 • 不同处理器可以单独运行但共享相同的内存资源 • 处理器对某内存地址的改变对其他所有处理器可见	• 分布式内存模型系统需要网络通信来连接各个处理器的内存 • 处理器拥有单独的本地内存和地址空间，并没有全局的地址空间 • 每个处理器单独运行，对其本地内存的改变对其他处理器没有影响 • 当处理器需要访问其他处理器的数据时，需要由程序来显式进行定义，并可能需要进行同步处理

Figure 4: 两种并行计算模型的特点



(a) 共享内存模型　　(b) 分布式内存模型

Figure 5: 并行计算的两种模型

也用作服务器与存储系统之间的直接或交换互连，以及存储系统之间的互连 [37]。

InfiniBand 高效率、低延迟，且能够支持一个子网中上千节点，并可以通过路由几乎无限扩展。同时，它支持 RDMA (Remote Direct Memory Access) 技术，能够绕开内核直接读取数据，从而有更高吞吐量和更低延迟 [18, 41, 21]。

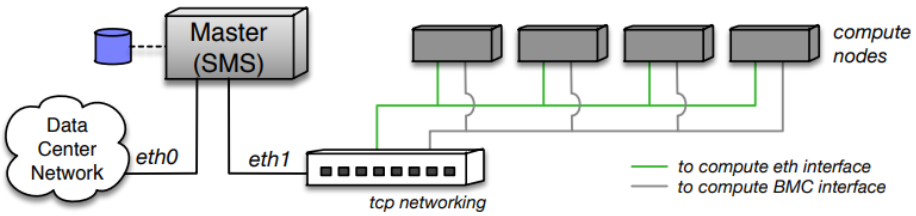


Figure 6: ohpc 示例的 HPC 集群的整体架构 [13]

2.4 HPC 技术栈

HPC 技术栈选择较多，主要包含操作系统、集群调度、编译器等涵盖 HPC 管理、编程、监控等各个维度的工具链。

2.4.1 操作系统

HPC 几乎全部使用 Linux 操作系统，但大多为 RH/SUSE 系列，例如劳伦斯 Livermore 国家实验室 (LLNL) 的超算使用的是给予 RHEL 的 TOSS(Trilab Operating system Software Stack)：

- Red Hat 系列，包括 RHEL、CentOS、Rocky Linux
- SUSE 系列，包括 SLES、Open SUSE

2.4.2 文件系统

HPC 集群中一些常用的文件系统包括:

- NFS: 中小型集群中使用较多
- GPFS: 高性能的专有并行文件系统
- Lustre: 高性能的开源并行文件系统

2.4.3 操作系统置备及集群管理

集群管理软件负责初始化以及维护裸金属服务器或者虚拟机，例如操作系统置备 (OS Provision)、固件更新、软件管理、文件分发等，对于大规模集群来说，这是十分有必要的。尽管这类软件是为 HPC 而开发的，但也可以用来管理其他类型的集群，例如 Kubernetes 集群。

主要有以下两种选择：

- Warewulf: 轻量级、无状态的管理软件
- xCAT: 比 Warewulf 功能更强大，但也更复杂一些。2018 Top500 排名第 1 和第二的 Summit、Sierra 均采取 xCAT 进行管理，并为 IBM 的官方 HPC 管理软件

2.4.4 资源管理/作业调度

分布式资源管理系统 (D-RMS) 负责控制 HPC 集群中硬件资源的使用，包括 CPU、内存、磁盘、带宽等，用户提交任务后，D-RMS 将选择以最优的方式进行任务分配和执行 [43, 3]，主要包括以下几种：

- Portable Batch Systems(PBS): OpenPBS 业内使用最广泛的集群调度系统，它还有商用版本 PBSPro
- SLURM: 另一款开源调度系统
- Platform Load Sharing Facility(LSF): IBM Spectrum LSF 社区版
- Grid Engine: 支持网格计算的资源和负载管理系统，最初由 Sun 创建 [40]

除了传统的 HPC 调度系统，Kubernetes 也是一种分布式资源管理和调度系统，但它从设计上与传统 HPC 调度系统有一定的差别，分别适合不同的场景 [8]。

2.4.5 编译器及开发工具

HPC 任务通常由 c/c++/fortran 编写，代码需要在目标平台上进行编译后执行，因此需要安装相应的编译器。一些商用编译器还特别对并行计算代码进行优化。主要有：

- GNU 编译器族: gcc/g++ 等
- llvm 编译器族: clang/clang++ 等
- ARM 平台 c/c++/fortran 编译器 [30]
- 商用编译器，如 Intel 编译器、PGI 编译器、PathScale 等
- 工具链: autoconf、automake、cmake、python 等
- 其他: Valgrind 内存调试器、Spack 包管理

2.4.6 性能分析工具

性能分析工具用来分析 HPC 应用的表现及调优，从而能够更好地利用资源。例如:

- BSC 工具集：Paraver、Dimemas、Extrae 等
- OSU Micro-benchmarks、gperftools、perf 等

2.4.7 开发库

HPC 应用编译及运行需要的一些软件库:

- 并行 I/O 库: 如 ADIOS、HDF5、NetCDF、PnetCDF 等
- 线性代数库: 如 PETSc、BLAS、Lapack、SuperLU 等
- 偏微分方程库: 如 PETSc、Trilinos 等
- 图算法库: 如 PBGL、Giraph 等
- 网格分解: 如 METIS、ParMETIS、Zoltan
- 可视化: 如 VTK、Gnuplot、Matplotlib、ParaView 等

- 消息传递: 分布式内存模型下需要，主要是 MPI[4] 包括 OpenMPI、MPICH、Intel MPI 等实现
- 多线程: 共享内存模式下需要，主要是 OpenMP、Pthreads 等

2.4.8 加速器相关

HPC 中需要用到 GPU 等进行加速，而 GPU 本身是为图形处理而设计的，需要使用一些框架来进行异构计算编程，主要包括:

- CUDA: 适用于 NVIDIA 显卡的专有 GPU 编程工具包
- OpenCL: 开放计算语言 (Open Computing Language)，是一个异构编程规范
- C++ AMP: C++ 加速大规模并行 (C++ Accelerated Massive Parallelism) 是由微软开发的 C++ 编译器和扩展集，使得 C++ 应用程序可以利用 GPU 等进行加速 [6]
- OpenACC: 开源加速器 (Open Accelerators) 是一个是为了简化异构计算（CPU/GPU）系统的并行编程的并行计算编程标准。

3 传统 HPC 与 AI、Big Data 的融合

3.1 HPC 与 AI 和 Big Data 融合的需求与可能性

传统 HPC 同 AI、Big Data 融合是新的发展趋势, 由此产生新的领域 HPDA(High Performance Data Analysis) [5]，旨在将 HPC、Big Data 以及 AI 进行融合产生更大的价值。他们之间的关系如 fig. 7所示。

之所以需要并存在这个可能性，主要的原因有 [46]:

[4]Message Passing Interface(MPI) 是一个跨语言的通讯协议，用于编写并行计算机。支持点对点和广播。MPI 是一个信息传递应用程序接口，包括协议和和语义说明，他们指明其如何在各种实现中发挥其特性。MPI 的目标是高性能，大规模性，和可移植性。MPI 在今天仍为高性能计算的主要模型 [38]。

[5]关于 HPDA 并没有明确的定义，多数表述为 HPDA = HPC + Big Data，也有将 AI 结合即 HPDA = HPC + Big Data + AI

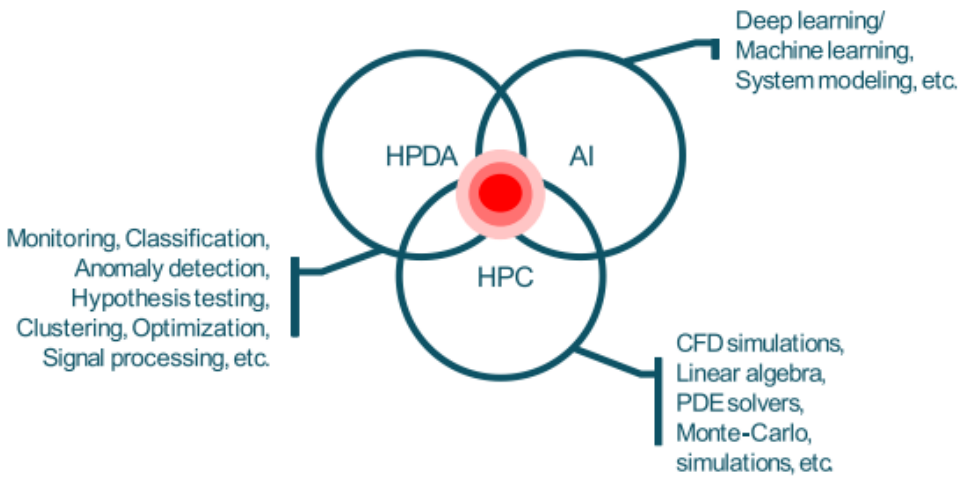


Figure 7: HPC 与 HPDA、AI 之间的关系

- 在科学领域，AI for Science 出现使得 HPC 程序中也包含 AI 算法，HPC+AI 是刚性需求
- 数据处理是 AI 的基础，数据与 AI 融合自然高效
- HPC 本身就存在大数据[6]
- HPC、AI、Big Data 在基础设施上有着类似的要求，如对内存、高性能网络和存储；对计算精度的要求上存在差别（HPC 要求 64 位浮点数，AI 只需要 16 位），如果能够在一定程度上复用那么在处理器层面可以同时支撑 HPC 和 AI。fig. 8直观反应了传统 HPC 与 HPDA 间的差异。

HPDA 的一个应用场景是将科研仿真和 AI 计算可以非常有效地进行联合，因为二者都需要模型和数据。通常，仿真使用（数学）模型产生数据，（AI）分析使用数据来生成模型。使用分析方法得到的模型和其他的模型一起可以被用到仿真中去；仿真产生的数据和其他来源的数据一起可以被用于分析。这样就形成了一个相互促进的良性循环 [45]。

[6] “Do you know what high-performance guys call big data?”…The answer: Data. [25]

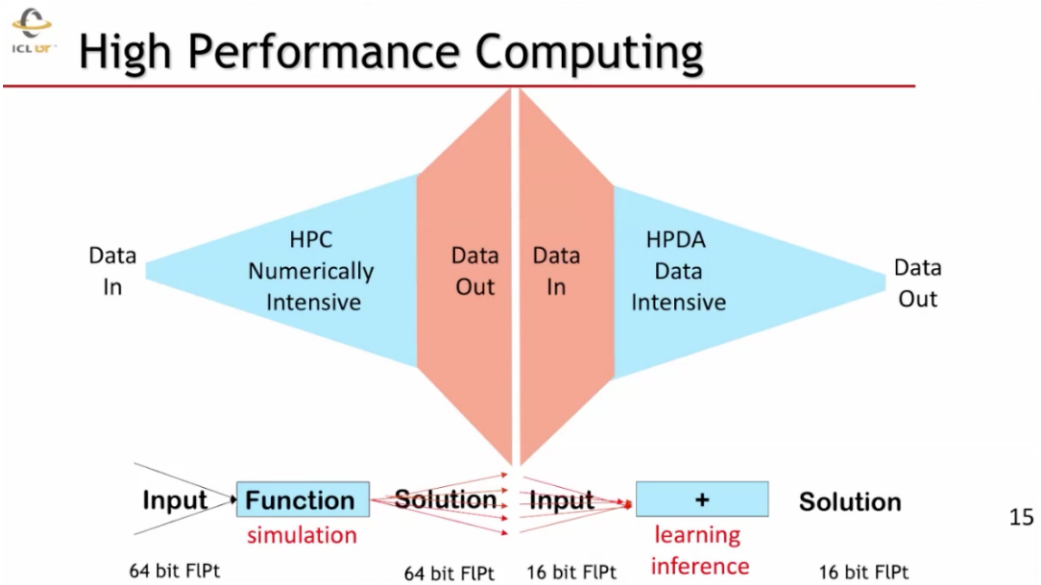


Figure 8: HPC 与 HPDA 的场景对比 [45]

在应用 HPDA 时，可以根据需要组合不同的形式的计算，从而能够端到端在一个系统内完成功能，例如使用 Sprak 对数据进行预处理，使用 MPI+TenserFlow 来训练模型等。

3.2 HPDA 融合存在的挑战

HPDA 目前处于较为前沿的阶段，尽管 HPC、AI、Big Data 等都发展了几十年近乎成熟，要应用 HPDA 仍然存在诸多的挑战，主要有：

- 技术栈不兼容，例如编程语言、框架等，相互之间存在较大的差别（参见 fig. 9），比如大数据使用的 Spark、Hadoop 等无法在 HPC 上使用 [25]
- 需要通用的计算架构，已经有一些将它们融合的尝试，例如将 Spark 应用在 HPC 上 [7]，但是缺乏通用性，且扩展性差
- 需要通用大容量/高速存储架构，HPC 与大数据采取的技术各不相同，HPC 多使用并行文件系统如 GPFS, Lustre；大数据多用 HDFS，需要设计更通用的架构 [42, 24]。

- 通用的资源管理/调度架构，HPC 更多是长时运行并行任务，而数据分析则更需要弹性伸缩，因此需要实现更通用的资源管理架构 [5, 26]

	HPC	Big Data (Hadoop/Spark)	AI
Primary Languages	C/C++/Fortran	Java/Python/Scala/R	Python/C++/R
Scaling	Strong	Weak	Both
Cluster Stack Management	Custom	Cloudera Manager/Apache/ Ambari/MapR Control System	NA
Schedulers	Slurm, PBS	YARN, Mesos	NA
File Systems	High-performance Distributed POSIX	Distributed Across Node Local Storage – HDFS	Posix or HDFS
OS	RedHat/CentOS SuSE	RedHat/CentOS SuSE	Ubuntu

Figure 9: HPC 与 AI、Big Data 的软件环境对比 [4]

在科研领域已经有开源的Ophidia HPDA 框架，尝试将 HPC 和 Big Data 结合起来，解决 eScience 领域的大数据问题。它的整体架构如 fig. 10所示 [11]。

3.3 HPDA 与 serverless

应用 HPDA 需要各种各样的编程环境和运行时，为了进行性能调优可能还需要内嵌调试信息，计算负载如何能够随时随地执行（尤其是虚拟化环境中），是一个关键问题。一种思路是将各个资源内嵌到工作负载的编译产物中，但是这样会带来很多无关代码、增大执行程序体积；另一种方式则是屏蔽掉底层细节，使用容器、甚至 serverless/FaaS 模式。

以容器化为代表的虚拟化技术发展如今已经十分成熟（参见 fig. 11），在应用领域已经应用十分广泛。利用容器化技术，可以将 HPDA 运行时进行封装，从而能够更容易进行部署和伸缩，极大提高应用的可移植性。例如 NVIDIA 提供 AI/HPC 容器[7]，每月更新提供更佳的性能优化。而 FaaS 能更好地提供逻辑抽象、数据模型等，因此有望作为更流畅的 HPC 运行时 [10]。

[7] 见:https://developer.nvidia.com/ai-hpc-containers

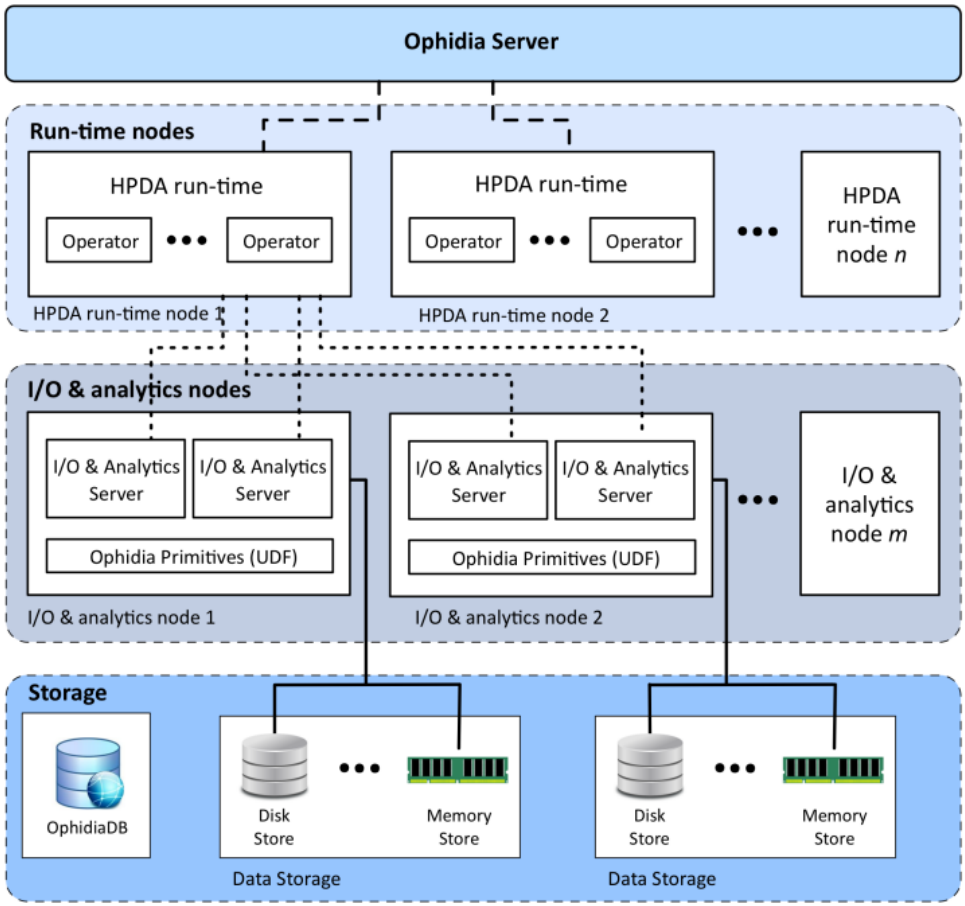


Figure 10: Ophidia HPDA 框架的顶层架构

4 社区及开源框架

4.1 相关社区

- Linux Foundation: Linux 基金会是一个非盈利性的联盟，其目的在于协调和推动 Linux 系统的发展，以及宣传、保护和规范 Linux
- Cloud Native Computing Foundation:Linux 基金会旗下的非盈利组织, 使命是创造和推动采用云原生计算来助力企业在云计算模式下更好的构建可扩展的

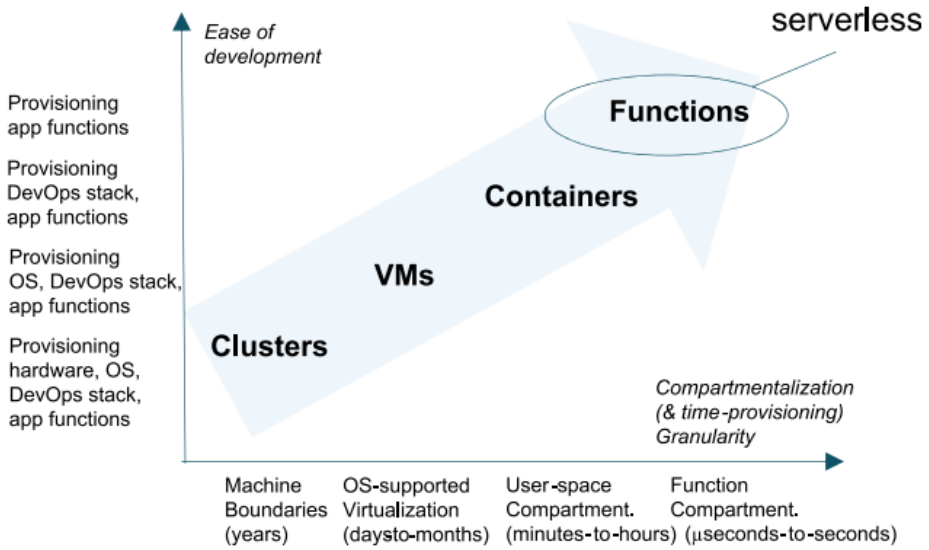


Figure 11: 虚拟化技术的演进 [10]

应用程序。

- OpenHPC: Linux 基金会旗下项目，关注于提供开源的 HPC 组件和最佳实践，以降低 HPC 使用门槛

4.2 HPC 相关框架

4.2.1 OpenHPC/ohpc

ohpc是 OpenHPC 社区维护的一套开源 HPC 集群部署指南，包含操作系统置备、资源管理、开发工具、运行时等各类 HPC 相关组件，

- 操作系统可选 Rocky8、Leap15
- 操作系统置备可选 Warewulf、xCAT[8]

[8]由于 xCAT 不支持 Rocky8，当前最新2.5版本不提供 xCAT 选项。

```python
def handle(event, context):
    return {
        "statusCode": 200,
        "body": "Hello from OpenFaaS!"
}
```

Figure 12: OpenFaaS 下使用 Python 编写的”Hello World”

- 资源管理可选 OpenPBS、SLURM
- 支持 x86_64、aarch64 两种架构

4.3 Serverless 框架

目前，已经出现不少的开源 serverless 框架，大多基于 Kubernetes 平台。在 serverless 平台下，用户只需要编写类似 fig. 12的函数来处理业务逻辑，由平台负责进行自动调度。

已知的 serverless 框架大致有以下几种 [28]:

框架	License	Contributers	Stars
OpenFaas	MIT	164	22.1k
Apache Openwhisk	Apache-2.0	200	5.8k
Knative	Apache-2.0	245	4.7k
~~Kubeless~~	Apache-2.0	105	6.9k
Fission	Apache-2.0	139	7.2k
~~Fn~~	Apache-2.0	84	5.2k
Nuclio	Apache-2.0	73	4.6k
~~Iron Functions~~	Apache-2.0	33	3k
~~OpenLambda~~	Apache-2.0	25	785

其中,

- Kubeless 的支持公司 VMware 已宣布停止维护 Kubeless，不建议继续使用
- Fn 已不活跃，最新版本仍为 2017 年发布，不建议使用
- Iron Functions 已不活跃，最新版本仍为 2018 年发布，不建议使用
- OpenLambda 仍处于初期，不具备 Production-ready，不建议使用

4.3.1 框架功能对比

框架	容器编排引擎	编程语言支持	内置监控系统
OpenFaaS	Kubernetes	NodeJs、Python、Go、Ruby、Java、C#	Prometheus、Grafana
Apache Openwhisk	Kubernetes	Go、Java、NodeJs、C#、PHP、Python、Ruby、Rust、Scala、Swift	N/A
Knative	Kubernetes	Any	Prometheus/ Open-Telemetry + Grafana
Fission	Kubernetes	NodeJs、Python、Go、Java、Ruby、PHP、C#	Prometheus、Open-Telemetry、Grafana、Linkerd
Nuclio	Kubernetes	Go、C、Python、Java、NodeJs	Prometheus、Azure Application Insights

4.3.2 框架性能对比

根据一些测试结果来看，Knative、Fission 以及 OpenFaaS 的性能受计算类型(计算密集型、I/O 密集型)、编程语言等的影响较大，不同场景下的性能差异较大[2, 23]。

4.3.3 OpenFaaS

OpenFaas是基于 Kubernetes 实现的 serverless 框架，支持 Java、Go、Python 等多种语言，用户只需要编写业务处理逻辑，由 OpenFaaS 包装为容器镜像执行，其执行流程如 fig. 13所示 [27]。

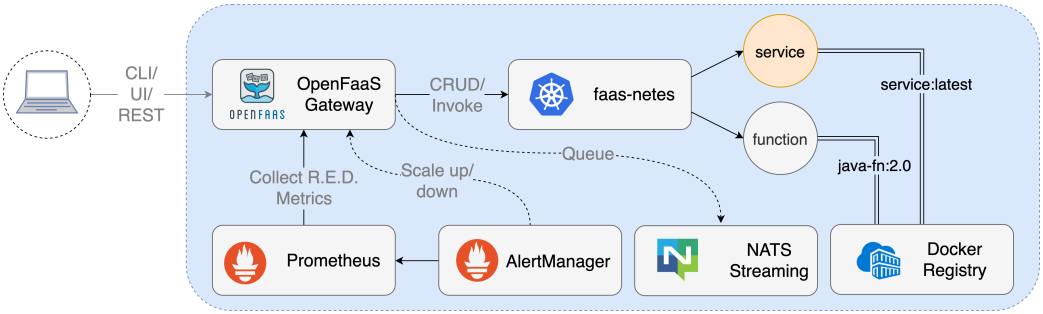


Figure 13: OpenFAAS 运行流程

它的架构如 fig. 14所示 [27]，主要包含：

- CI/GitOps 层: 通过 CI 将用户编写的代码并部署到集群中
- 应用层: 包含网关、NATS 和 Prometheus，其中网关暴露 REST 形式的管理功能，NATS 用于异步执行，Prometheus 管理监控数据
- 基础层: 基于 Kubernetes 的运行时

4.3.4 Knative

Knative是由谷歌开源的一款 serverless 框架，是 CNCF 的孵化项目。Knative 是基于 Kubernetes 的 CRD(Kubernetes Custom Resource Definitions) 和 Istio 实现的纯原生的架构，包含以下的 CRD:

- Service: 负责管理整个计算负载的生命周期，控制其他资源的创建
- Route: 将网络映射到 revision
- Configuration: 负责管理配置信息，更新配置时，会创建新的 revision

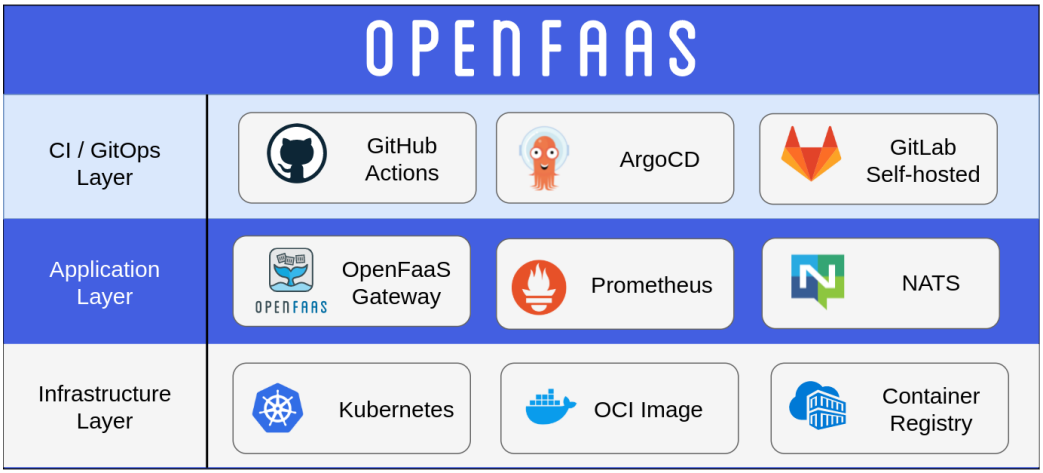


Figure 14: OpenFAAS 架构概览

- Revision: 计算负载的一个特定的版本实例（包含配置和代码等），能够根据负载自动进行伸缩

整个架构如 fig. 15所示 [22]。

4.3.5 Nuclio

Nuclio是一个开源的用户科学计算的 serverless 框架，号称最快的 serverless 框架[9]，并支持 GPU 加速。Nuclio 开发的初衷是现有的 serverless 框架不满足以下几点 [32]:

- 最小化 CPU/GPU 以及 I/O 使用、最大化并行度
- 原生能够支持多种数据源、触发方式、处理模型以及 ML 框架
- 有状态的函数处理（包含路径加速）

[9]"Real-time performance running up to 400,000 function invocations per second".

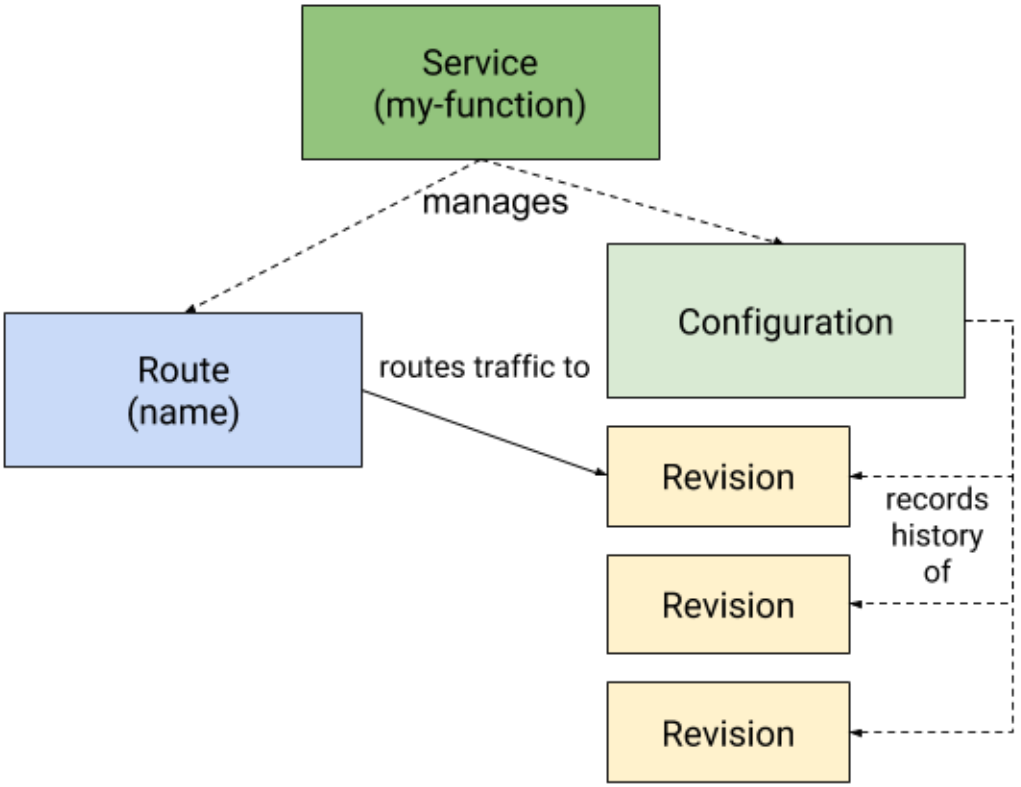


Figure 15: Knative 的模型

5 总结

本文对 HPC 相关的历史、应用以及技术原理和框架进行了一个概要的回顾，着重学习了 HPC 的发展趋势、以及其与 AI、Big Data 相关技术融合的相关资料。结合我司的实际情况，得出以下的结论：

- 传统 HPC 应用场景偏向科研、学术研究等，在我司缺乏应用场景
- HPDA 是 HPC 的新应用领域和发展趋势，具备可行性并处于发展阶段
- 传统 HPC 领域更多在探讨如何在现有 HPC 集群上利用 AI/Big Data 技术；对我司来说需要借鉴 HPC 来更好的实现 AI/Big Data，容器化 +serverless 是更具有战略价值的方向

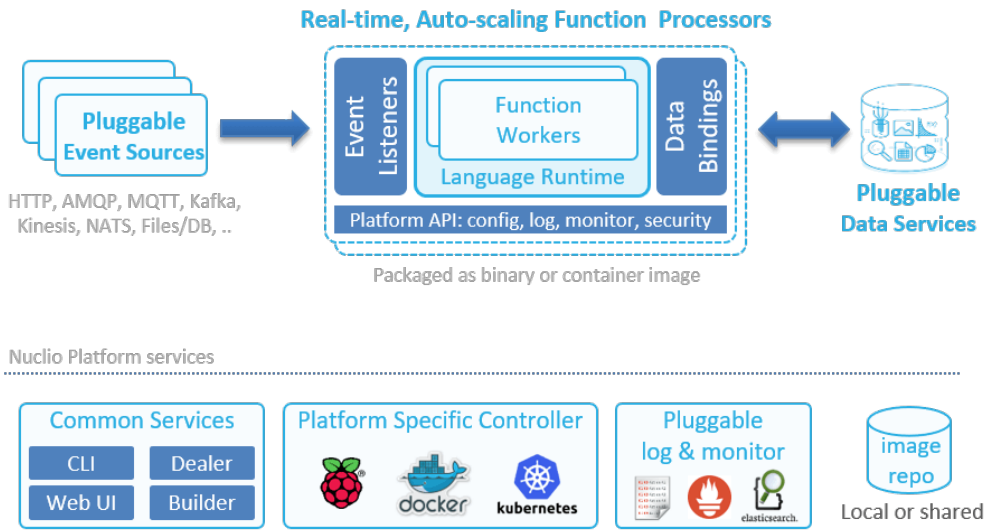


Figure 16: Nuclio 的整体架构

References

[1] AMD. 高性能计算 *(HPC)*. [Online; accessed 13-September-2022].
URL: https://www.amd.com/zh-hans/technologies/hpc-explained.

[2] David Balla, Markosz Maliosz, and Csaba Simon.
"Open source faas performance aspects". In: *2020 43rd International Conference on Telecommunications and Signal Processing (TSP)*. IEEE. 2020, pp. 358–364.

[3] *Batch-Scheduler*. [Online; accessed 15-September-2022]. 2022.
URL: https://hpc-wiki.info/hpc/Batch-Scheduler.

[4] *Bringing AI Into Your AGENCY HPC Environment*.
[Online; accessed 16-September-2022].
URL: https://www.govexec.com/media/intel_ai-hpc_eguide.pdf.

[5] Chansup Byun et al.
"LLMapReduce: Multi-level map-reduce for high performance data analysis".
In: *2016 IEEE High Performance Extreme Computing Conference (HPEC)*.
IEEE. 2016, pp. 1–8.

[6] *C++ AMP Overview*. [Online; accessed 15-September-2022]. 2021.
URL: https://docs.microsoft.com/en-us/cpp/parallel/amp/cpp-amp-overview?view=msvc-170.

[7] Nicholas Chaimov et al. "Scaling spark on hpc systems".
In: *Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing*. 2016, pp. 97–110.

[8] *Choosing the Right Scheduler for HPC and AI Workloads*.
[Online; accessed 15-September-2022]. 2019.
URL: https://www.hpcwire.com/solution_content/ibm/cross-industry/choosing-the-right-scheduler-for-hpc-and-ai-workloads/.

[9] Lars Cromley. *What is High Performance Computing (HPC)*.
[Online; accessed 13-September-2022]. 2017.
URL: https://www.2ndwatch.com/blog/introduction-to-high-performance-computing-hpc/.

[10] Nicolas Dube et al. "Future of HPC: The Internet of Workflows".
In: *IEEE Internet Computing* 25.5 (2021), pp. 26–34.

[11] Donatello Elia, Sandro Fiore, and Giovanni Aloisio.
"Towards HPC and Big Data Analytics Convergence: Design and Experimental Evaluation of a HPDA Framework for eScience at Scale".
In: *IEEE Access* 9 (2021), pp. 73307–73326.

[12] Rob Farber. "AI-HPC is Happening Now".
In: *insideHPC Special Report, insideHPC, LLC* (2017).

[13] hhpc. *OpenHPC (v2.5) Cluster Building Recipes*.
[Online; accessed 15-September-2022]. 2022.
URL: https://github.com/openhpc/ohpc/releases/tag/v2.5.GA.

[14] *High-performance comHistory and overview of high performance computing*.
CPS343. 2020. URL: https://www.math-cs.gordon.edu/courses/cps343/presentations/History_and_Overview_of_HPC.pdf.

[15] *HPC and AI: A Powerful Combination*. [Online; accessed 15-September-2022].
2022. URL: https://www.intel.com/content/www/us/en/high-performance-computing/hpc-artificial-intelligence.html.

[16] *HPC system designs*. [Online; accessed 13-September-2022].
URL: https://www.ibm.com/topics/hpc#:~:text=High%2Dperformance%20computing%20(HPC),complex%20problems%20requiring%20massive%20computation..

[17] hpda. *High Performance Data Analytics*. [Online; accessed 13-September-2022].
URL: https://www.heavy.ai/technical-glossary/high-performance-data-analytics.

[18] *InfiniBand: The Future Standard of High-Performance Computing*. 2022.
URL: https://www.nextrongroup.com/News_detail/121/InfiniBand:-The-Future-Standard-of-High%E2%80%94Performance-Computing.html.

[19] *Introduction to Cluster Computing*. [Online; accessed 13-September-2022].
URL: http://selkie.macalester.edu/csinparallel/modules/DistributedMemoryProgramming/build/html/IntroCluster/IntroCluster.html.

[20] E Joseph et al.
"HYPERION RESEARCH UPDATE: Research Highlights In HPC, HPDA-AI,

Cloud Computing, Quantum Computing, and Innovation Award Winners".
In: *SC19* (2019).

[21] Zeus Kerravala. *Despite Predictions of Its Demise, InfiniBand Is Still Alive*. 2020. URL: https://www.eweek.com/networking/despite-predictions-of-its-demise-infiniband-is-still-alive/#:~:text=Today%2C%20both%20InfiniBand%20and%20Ethernet,up%20to%2012%20SerDes%20together)..

[22] *Knative Concepts*. URL: https://knative.dev/docs/concepts/.

[23] Junfeng Li et al. "Understanding open source serverless platforms: Design considerations and performance".
In: *Proceedings of the 5th international workshop on serverless computing*. 2019, pp. 37–42.

[24] Jianwei Liao et al. "A flexible I/O arbitration framework for netCDF-based big data processing workflows on high-end supercomputers".
In: *Concurrency and Computation: Practice and Experience* 29.15 (2017), e4161.

[25] Bill Magro et al.
High Performance Data Analytics: A Q&A with Subject Matter Experts. 2017.
URL: https://www.cio.com/article/230297/high-performance-data-analytics-a-qanda-with-subject-matter-experts.html.

[26] Michael Mercier et al.
"Big data and HPC collocation: Using HPC idle resources for Big Data analytics".
In: *2017 IEEE International Conference on Big Data (Big Data)*. IEEE. 2017, pp. 347–352.

[27] *OpenFaaS stack*.
URL: https://docs.openfaas.com/architecture/stack/.

[28] Andrei Palade, Aqeel Kazmi, and Siobhán Clarke. "An evaluation of open source serverless computing frameworks support at the edge".
In: *2019 IEEE World Congress on Services (SERVICES)*. Vol. 2642. IEEE. 2019, pp. 206–211.

[29] Thomas Sterling, Maciej Brodowicz, and Matthew Anderson.
High performance computing: modern systems and practices.
Morgan Kaufmann, 2017.

[30] *Tools and Software:HPC*. [Online; accessed 15-September-2022]. 2022. URL: https://developer.arm.com/products/software-development-tools/hpc.

[31] *When Amdahl's Law Doesn't Apply*. [Online; accessed 14-September-2022]. URL: https://insidehpc.com/2011/06/21047/.

[32] *Why another "serverless" project?* URL: https://nuclio.io/docs/latest/introduction/.

[33] Wikipedia contributors. *Amdahl's law — Wikipedia, The Free Encyclopedia*. [Online; accessed 14-September-2022]. 2022. URL: https://en.wikipedia.org/w/index.php?title=Amdahl%27s_law&oldid=1108054797.

[34] Wikipedia contributors. *FLOPS — Wikipedia, The Free Encyclopedia*. [Online; accessed 9-September-2022]. 2022. URL: https://en.wikipedia.org/w/index.php?title=FLOPS&oldid=1105186915.

[35] Wikipedia contributors. *Gustafson's law — Wikipedia, The Free Encyclopedia*. [Online; accessed 14-September-2022]. 2022. URL: https://en.wikipedia.org/w/index.php?title=Gustafson%27s_law&oldid=1107600217.

[36] Wikipedia contributors. *High-performance computing — Wikipedia, The Free Encyclopedia*. [Online; accessed 7-September-2022]. 2022. URL: https://en.wikipedia.org/w/index.php?title=High-performance_computing&oldid=1106282408.

[37] Wikipedia contributors. *InfiniBand — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=InfiniBand&oldid=1106147193. [Online; accessed 21-September-2022]. 2022.

[38] Wikipedia contributors. *Message Passing Interface — Wikipedia, The Free Encyclopedia*. [Online; accessed 15-September-2022]. 2022. URL: https://en.wikipedia.org/w/index.php?title=Message_Passing_Interface&oldid=1104876573.

[39] Wikipedia contributors. *Omni-Path — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Omni-Path&oldid=1046427810. [Online; accessed 21-September-2022]. 2021.

[40] Wikipedia contributors.
Oracle Grid Engine — Wikipedia, The Free Encyclopedia.
[Online; accessed 15-September-2022]. 2022. URL: https://en.wikipedia.org/w/index.php?title=Oracle_Grid_Engine&oldid=1066808425.

[41] Wikipedia contributors.
Remote direct memory access — Wikipedia, The Free Encyclopedia.
https://en.wikipedia.org/w/index.php?title=Remote_direct_memory_access&oldid=1093871583. [Online; accessed 21-September-2022]. 2022.

[42] Pengfei Xuan et al.
"Big data analytics on traditional HPC infrastructure using two-level storage".
In: *Proceedings of the 2015 International Workshop on Data-Intensive Scalable Computing Systems*. 2015, pp. 1–8.

[43] Yonghong Yan and Barbara Chapman. "Comparative study of distributed resource management systems–sge, lsf, pbs pro, and loadleveler".
In: *Technical Report-Citeseerx* (2008).

[44] 什么是高性能计算（*HPC*）?. [Online; accessed 13-September-2022].
URL: https://www.redhat.com/zh/topics/high-performance-computing/what-is-high-performance-computing.

[45] 图灵奖得主 *Jack Dongarra*：高性能计算与 *AI* 大融合，如何颠覆科学计算.
[Online; accessed 15-September-2022]. 2022.
URL: https://www.51cto.com/article/717823.html.

[46] 郑纬民. 关于算力的几点考虑. YEF 2022 特邀报告（6 月 9 日）.
URL: https://dl.ccf.org.cn/video/videoDetail.html?_ack=1&id=5992273244899328.

[47] 高性能计算 *(HPC)* 是什么?. [Online; accessed 13-September-2022].
URL: https://www.oracle.com/cn/cloud/hpc/what-is-hpc/.