LSF调研报告

范翔宇 ad2018@mail.ustc.edu.cn

第一章 调研背景

1.1 HPC简介

1.1.1 高性能计算

高性能计算简称 HPC，是指利用聚集起来的计算能力来处理标准工作站无法完成的数据密集型计算任务，包括仿真、建模和渲染等。我们在处理各种计算问题时常常遇到这样的情况：由于需要大量的运算，一台通用的计算机无法在合理的时间内完成工作，或者由于所需的数据量过大而可用的资源有限，导致根本无法执行计算。HPC 方法通过使用专门或高端的硬件，或是将多个单元的计算能力进行整合，能够有效地克服这些限制。将数据和运算相应地分布到多个单元中，这就需要引入并行 概念。

就硬件配置而言，常用的类型有两种：共享内存计算机和分布式内存集群。在共享内存计算机上，所有处理单元都可以访问随机存取存储器（RAM）；而在分布式内存集群中，不同的处理单元或节点 之间无法访问内存。在使用分布式内存配置时，由于不同的处理单元不能访问同一个内存空间，因此必须存在一个相互连接的网络，才能在这些单元之间发送消息（或者使用其他通信机制）。鉴于有些单元共享共同的内存空间，而其他单元又是另一种情况，现代 HPC 系统通常是融合了这两个概念的混合体。

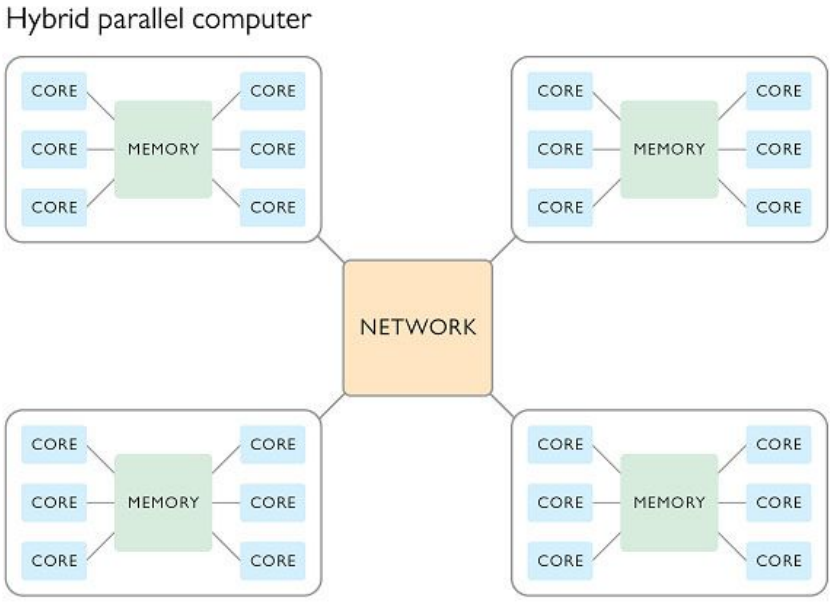

图1:混合并行计算[1]

1.1.2 使用HPC的原因

首先，随着中央处理单元（CPU）和节点数量的不断增加，人们可以使用的计算能力越来越多。有了强大的计算能力，就能在单位时间内执行更多运算，从而提高特定模型的计算速度，即加速比。但是由Amdahl定律，我们知道加速比是有限的。另一方面，在集群的情况下，可用的内存量通常以线性方式增加，同时包含更多的节点。这样，随着计算单元数量的增加，就能够处理越来越大的模型。从某种意义上来说，运用这种方法可以对Amdahl定律提出的限制加以“欺骗”——该定律适用于固定大小的计算问题。将计算能力和内存提高一倍，就可以在相同的时间内完成大小为基本任务两倍的计算任务。

1.2 LSF简介

随着数据中心在规模和复杂性上的快速增加，使得对集群工作负载和应用的管理更加困难，同时也难以确保计算硬件和软件等资源的正常使用。用户希望在任何地方都能灵活使用应用程序，并自动操控数据流，管理者希望能够监督集群的资源和负载，管理软件使用权，确定瓶颈问题，监督面向用户的服务协议是否满足，并计划系统的扩容数量。

IBM Spectrum LSF(最初为Platform Load Sharing Facility)平台系列软件正致力于解决以上问题，对于要求高的分布式关键任务型高性能计算环境，LSF软件产品提供一个高性能负载管理平台(支持Linux和Windows)，这个平台有一套综合的基于智能的，策略驱动的调度策略，方便用户使用所有运算基础设施资源，帮助保障最佳的应用性能。

LSF平台管理批处理负载，它将一个分布式计算网络平台作为一个超级计算机，将用户对资源的请求匹配，这个平台智能地将合适的工作分给正确的资源，使资源有效利用，减少浪费并实现最佳性能。

第二章 核心内容[2][3]

本章主要介绍LSF的运行过程和调度策略，这也正是我们重点关注的。在此之前，我们需要对部分缩写进行解释：mbatchd —— Master Batch Daemon running on the master host，后台的主进程；mbschd —— Master Batch Scheduler Daemon running on the master host，后台的主调度进程；sbatchd —— Slave Batch Daemon running on each server host，从属批处理进程。

2.1 运行过程

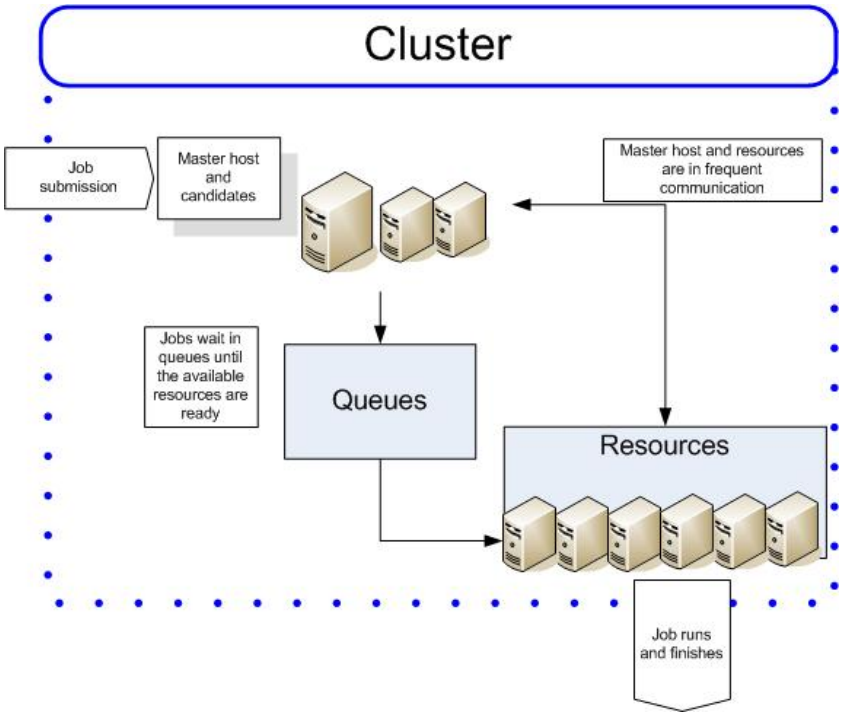

图2:LSF总览[2]

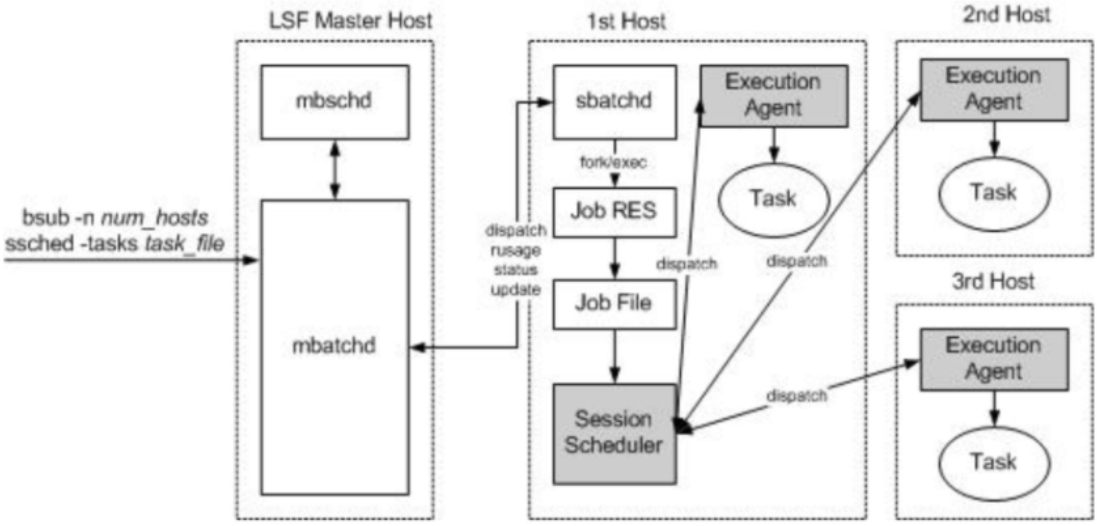

图3:LSF作业调度结构[2]

2.1.1 提交作业

从LSF客户端，或者是一个运行bsub命令的服务器上提交一份作业，当提交这份作业时，如果不指定哪个队列，这份作业就会被提交到系统默认的队列中，作业在队列中等待安排，这些作业处于等待状态。

2.1.2 调度作业

后台的主进程mbatchd将处理队列中的作业，在一个预定的时间间隔里将这些作业按设定的计划，传递给主调度进程mbschd。主调度进程mbschd评估这份工作时，根据作业的优先权制定调度决策、调度机制和可利用资源。主调度进程选择最佳的主机，在哪里作业可以运行，并将它的决策返回给后台主进程mbatchd。主负载信息管理进程(LIM)收集资源信息，主LIM与mbatchd主进程交流这些信息，mbatchd主进程通过这些交流信息支持调度决定。

2.1.3 分配作业

mbatchd主进程一收到mbschd发过来的决定，立即分配作业到对应主机。

2.1.4 运行作业

从属批处理进程sbatchd，从mbatchd主进程接到要求，为这份作业创建一个子sbatchd和一个执行环境，通过使用一个远程执行服务器开始这个作业。

2.1.5 返回输出

当一个作业完成时，如果这个作业没有任何问题，它处于一个完成状态。如果有错误作业无法完成，这份作业处于退出状态。sbatchd传达作业信息，包括错误提示和给mbatchd的输出信息。

2.1.6 通知用户

mbatchd通过邮件给提交主机反馈作业输出信息、作业错误、提示信息、作业信息。

2.2 调度策略

2.2.1 先来先调度策略

不需要任何配置，队列的默认行为就是先来先服务。

2.2.2 基于优先级的调度策略

管理员可以通过命令行改变作业的调度顺序。也可以考虑采用更灵活的方式，给予用户权利，提交作业的时候，设置作业优先级。

2.2.3 服务协议满足调度策略*

LSF 中的 SLA 是一种“即时”调度策略，用于调度 LSF 管理员和 LSF 用户之间商定的服务。SLA 调度策略定义应该从每个 SLA 运行多少作业以满足配置的目标。

2.2.4 公平共享策略

可以为不同的用户分组，配置不同的资源权重，从而获取相应权重的资源。它以队列为单位进行配置。为了更好实现公平，每一个用户，会根据自己的权重，以及资源的使用状况，计算出一个优先级。作业会根据优先级获取资源执行，最终达到优先级趋同，资源公平分配的目的。

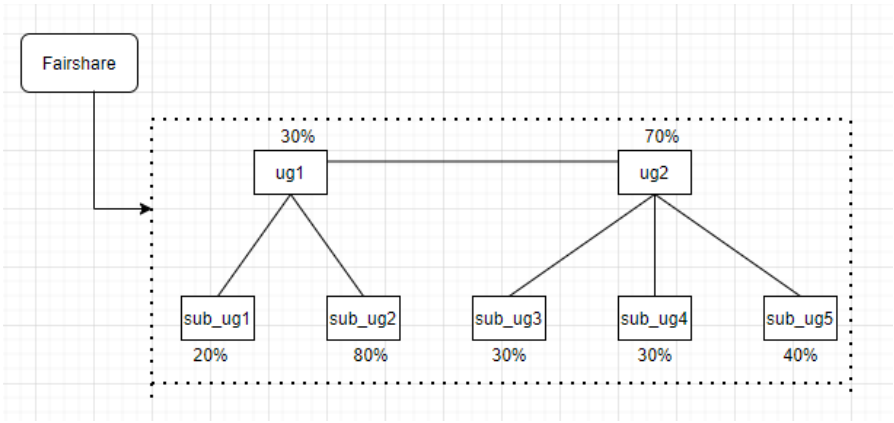

图4:公平共享

最简单的优先级公式可以理解为：优先级 = 权重 / (1 + used slots)。

LSF优先级公式为：

$$dynamicpriority = number_shares/(cpu_time * CPU_TIME_FACTOR + run_time * RUN_TIME_FACTOR + (1 + job_slots) * RUN_JOB_FACTOR + (1 + fwd_job_slots) * FWD_JOB_FACTOR + fairshare_adjustment * FAIRSHARE_ADJUSTMENT_FACTO ((historical_gpu_run_time + gpu_run_time) * ngpus_physical) * GPU_RUN_TIME_FACTOR$$

2.2.5 抢占

抢占是基于队列的，高优先级队列可以配置为抢占队列。当可抢占队列有作业运行的时候，高优先级队列将抢占这些作业的资源，来运行自己的作业。

2.2.6 限制

配置 `sub_ug1` 的限制，比如，最多占用6个slot，那么属于这个子用户组的用户提交的作业，最多占用6个slots运行，剩下的作业要 `PEND` 起来，等待同用户分组已经运行作业的结束，释放资源，才能获得资源使用的配额。

2.2.7 保留

为 `ug2` 创建保留资源池，属于保留池中的资源是为 `ug2` 保留的最低限度资源，不可以被别人使用。和限制一样，保留资源池也可以为不同的资源使用者保留多种资源。

2.2.8 预定

调度时为指定作业预留资源，如下例：

- 普通作业1：1 slot
- 指定作业2：2 slots
- 普通作业3：1 slot
- 普通作业4：1 slot

我们假设可用的slot只有两个，且作业运行时间一致。那么开始调度这三个作业时，如果没有预定机制，普通作业1分配到1个slot后，剩余1个slot，便调度普通作业3利用这个剩余的slot。普通作业1运行完毕后，普通作业3仍在运行，便调度普通作业4利用这个slot。只有普通作业1、3、4运行完后，指定作业2才能正常运行。最坏情况下，可能导致指定作业2饿死。但是如果引进预定机制后，在普通作业1分配到1个slot后，剩余1个slot被作业2预定了，不能被其他作业调度。

2.2.9 回填

仅仅预定还是会浪费slot。当保留的slot时间超过低优先级短作业的时间时候，可以将低优先级作业回填到预定的slot。不过需要比较准确的作业运行时间预测。

2.2.10 提前预定

管理员可以按需设置资源的提前预定。按照计划，为将来的作业预定资源。如：在今天上午10点到11点中，管理员需要执行管理性质作业，需要选择内存1G的计算节点，总计需要4个slot。注意与2.2.8预定区分开来。

第三章 评估指标与比较

本章主要介绍如何评估及比较LSF和slurm。

3.1 评估指标

首先，开源与否决定了我们在平台本身架构和作业调度上做研究的难易程度。同时还要考虑是否与其他HPC框架容易集成，吞吐量、资源利用率也都是我们的重点关注，因为其不仅直观体现了平台调度和作业工作的效率，还与平台框架和并行性密切相关。除此之外，我们还要考虑调度策略和资源共享程度。最后，平台的鲁棒性和风险大小也会影响到作业的正常运行。当然，以上只是简单陈列下应该要考虑到的指标，至于指标的权重，仍待后续讨论。

3.2 比较

从网络上找到了几张对比二者的图：

调度器	Spectrum LSF
全称	Load Sharing Facility
最新版本	Version 10.1.0
厂商	IBM
授权许可	商用
支持平台	Linux，Windows
Cloud Bursting溢出到云	通过LSF resource connector支持溢出到云
支持云厂商	AWS/Azure/Google Cloud
Auto-Scale自动伸缩	支持
云端费用监控	无
可视化界面	有
最大节点数	6k+
费用	付费
商业支持	国内有
风险	/

图5：Spectrum LSF基本情况[4]

调度器	Slurm
全称	Simple Linux Utility for Resource Management
最新版本	Version 20.02
厂商	SchedMD
授权许可	开源，目前由社区和 SchedMD 公司共同维护，保持开源和免费，由 SchedMD 公司提供商业支持
支持平台	Linux
Cloud Bursting溢出到云	Slurm的Cloud scheduling支持云上开关机，不支持创建集群。fastone支持
支持云厂商	fastone支持AWS/阿里云/Azure/腾讯云/华为云/Google Cloud
Auto-Scale自动伸缩	fastone支持
云端费用监控	fastone支持
可视化界面	fastone支持
最大节点数	120k+
费用	免费
商业支持	有，国内应该没有
风险	/
其他	60%全球TOP500超算中心和超大规模的集群包括我国的天河二号等都采用Slurm作为调度系统

图6：Slurm基本情况[4]

Competitor	Industry Focus	Competitor Disadvantage	Platform Computing Advantage
Open Source (SLURM Torque Grid Engine)	• Life Sciences • Education • Gov't	• Increased administrative burden • Lower throughput & utilization • No technical roadmaps makes planning a challenge & a risk	• Complete, certified & production-ready solutions • Large, global support organization • Reduced risk
Adaptive Computing (Moab)	• Industrial Mfg • Oil and Gas • Gov't	• Small community = limited best practices • Limited support & development organization • Limited stack of products that do not natively or coherently work together	• Installed in all HPC industries = robust set of best practices • Large, experienced support organization • Complete set of HPC solutions, including a robust HPC cloud roadmap
Altair Engineering (PBSPro)	• Industrial Mfg • Oil and Gas • Gov't	• Just a scheduler • Complicated products lack applicability outside of Tier I customers	• Robust, policy-based workflow management • Complete set of HPC solutions • Integrated application interface & support
Univa (UGE)	• Education • Life Sciences • Electronics	• Small organization w/ no ability to support large enterprises • Limited technology footprint & roadmap • Limited support install base	• Robust, competitively priced support • Complete, production-ready solutions • Robust set of admin tools

图7：LSF与其他调度平台的比较[5]

Slurm	• The most popular scheduler for managing distributed, batch-oriented HPC workloads • Integrates well with common HPC frameworks • Complex to use and maintain, particularly with containerized workloads
LSF	• Built for running diverse, finite, distributed workloads with flexible resource sharing • Sensitive to factors like timeliness, affinity, and topology • Complex to use and maintain, particularly with containerized workloads

图8：LSF与Slurm的比较[6]

总得来说，还是用slurm比较好，大多超算系统都在用slurm，而且还是开源免费，与普通HPC框架很好集成，我们或许可以在slurm上做一些更深层面的研究，但相对来说，slurm的吞吐量和利用率可能低了一点并且没有产品路线图。如果我们对调度和资源共享要求比较高，如需要考虑及时性、亲和力和拓扑结构等因素，可能应该考虑一下付费、风险较低的LSF。

第四章 相关领域和可能的后续工作方向

本章主要介绍相关领域的研究工作，为我们后续研究提供参考

4.1 作业运行时间预测

已有工作通过机器学习等结合作业的输入参数、历史日志[7]或者对作业代码进行插桩[8]等来进行作业运行时间预测。但是准确率并不是特别高，而且大多数都仅限于VASP。后续或许可以尝试其他模型来进行时间预测，或者尝试从作业的其他特征来切入，当然也可以从VASP扩展到其他作业。总得来说，时间预测可做的东西比较多的。

4.2 作业调度策略优化

已有工作通过深度强化学习与平台不断交互，来实现优化调度[9]。我们可以考虑用不同的方法来时间预测或者用不同的模型来与平台交互。同时，也要考虑调度策略的可移植性与兼容性。除此之外，我们还可以对已有的调度进行部分规则修改，如优化优先级计算公式使其针对性更强。总之，我们想提出一个更优的调度算法。

4.3 平台架构优化

我们可以分析平台架构，修改代码，搭建新颖的平台plus来优化解决特定问题。不过这个方向的工作量可能很大，短期内很难完成。

附：参考资料

[1]http://cn.comsol.com/blogs/hybrid-parallel-computing-speeds-up-physics-simulations/

[2]https://www.ibm.com/docs/en/spectrum-lsf/10.1.0

[3]http://scc.ustc.edu.cn/

[4]https://zhuanlan.zhihu.com/p/274276759

[5]https://blog.csdn.net/qq_43653083/article/details/120053551

[6]https://www.run.ai/guides/slurm-deep-learning/slurm-vs-lsf-vs-kubernetes-scheduler-which-is-right-for-you/

[7]A Novel Two-Step Job Runtime Estimation Method Based on Input Parameters in HPC System

[8]Automated Performance Modeling of HPC Applications Using Machine Learning

[9]RLSchert: An HPC Job Scheduler Using Deep Reinforcement Learning and Remaining Time Prediction