理解LoadAverage做好压力测试Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/644/2021_2022__E7_90_86_E8_A7_A3Load_c104_644865.htm SIP的第四期结束了，因为控制策略的丰富，早先的的压力测试结果已经无法反映在高并发和高压力下SIP的运行状况，因此需要重新作压力测试。跟在测试人员后面做了快一周的压力测试，压力测试的报告也正式出炉，本来也就算是告一段落，但第二天测试人员说要修改报告，由于这次作压力测试的同学是第一次作，有一个指标没有注意，因此需要修改几个测试结果。那个没有注意的指标就是load average，他和我一样开始只是注意了CPU，内存的使用状况，而没有太注意这个指标，这个指标与他们通常的限制（10左右）有差别。重新测试的结果由于这个指标被要求压低，最后的报告显然不如原来的好看。自己也没有深入过压力测试，但是觉得不搞明白对将来机器配置和扩容都会有影响，因此去问了DBA和SA，得到的结果相差很大，看来不得不自己去找找问题的根本所在了。 通过下面的几个部分的了解，可以一步一步的找出Load Average在压力测试中真正的作用。 CPU时间片 为了提高程序执行效率，大家在很多应用中都采用了多线程模式，这样可以将原来的序列化执行变为并行执行，任务的分解以及并行执行能够极大地提高程序的运行效率。但这都是代码级别的表现，而硬件是如何支持的呢？那就要靠CPU的时间片模式来说明这一切。程序的任何指令的执行往往都会要竞争CPU这个最宝贵的资源，不论你的程序分成了多少个线程去执行不同的任务，他们都必须排队等待获取这个资源来计算和处理命令。先看看

单CPU的情况。 任何线程如果都排队等待CPU资源的获取，那么所谓的多线程就没有任何实际意义。CPU Manager只是我虚拟的一个角色，由它来分配和管理CPU的使用状况，此时多线程将会在运行过程中都有机会得到CPU资源，也真正实现了在单CPU的情况下实现多线程并行处理。 多CPU的情况只是单CPU的扩展，当所有的CPU都满负荷运作的时候，就会对每一个CPU采用时间片的方式来提高效率。 在Linux的内核处理过程中，每一个进程默认会有一个固定的时间片来执行命令（默认为1/100秒），这段时间内进程被分配到CPU，然后独占使用。如果使用完，同时未到时间片的规定时间，那么就主动放弃CPU的占用，如果到时间片尚未完成工作，那么CPU的使用权也会被收回，进程将会被中断挂起等待下一个时间片。 CPU利用率和Load Average的区别 压力测试不仅需要对业务场景的并发用户等压力参数作模拟，同时也需要在压力测试过程中随时关注机器的性能情况，来确保压力测试的有效性。当服务器长期处于一种超负荷的情况下运行，所能接收的压力并不是我们所认为的可接受的压力。就好比项目经理在给一个人估工作量的时候，每天都让这个人工作12个小时，那么所制定的项目计划就不是一个合理的计划，那个人迟早会垮掉，而影响整体的项目进度。 CPU利用率在过去常常被我们这些外行认为是判断机器是否已经到了满负荷的一个标准，看到50%-60%的使用率就认为机器就已经压到了临界了。CPU利用率，顾名思义就是对于CPU的使用状况，这是对一个时间段内CPU使用状况的统计，通过这个指标可以看出在某一个时间段内CPU被占用的情况，如果被占用时间很高，那么就需要考虑CPU是否已经处于超负荷

运作，长期超负荷运作对于机器本身来说是一种损害，因此必须将CPU的利用率控制在一定的比例下，以保证机器的正常运作。 Load Average是CPU的Load，它所包含的信息不是CPU的使用率状况，而是在一段时间内CPU正在处理以及等待CPU处理的进程数之和的统计信息，也就是CPU使用队列的长度的统计信息。为什么要统计这个信息，这个信息的对于压力测试的影响究竟是怎么样的，那就通过一个类比来解释CPU利用率和Load Average的区别以及对于压力测试的指导意义。 我们将CPU就类比为电话亭，每一个进程都是一个需要打电话的人。现在一共有4个电话亭（就好比我们的机器有4核），有10个人需要打电话。现在使用电话的规则是管理员会按照顺序给每一个人轮流分配1分钟的使用电话时间，如果使用者在1分钟内使用完毕，那么可以立刻将电话使用权返还给管理员，如果到了1分钟电话使用者还没有使用完毕，那么需要重新排队，等待再次分配使用。 对于使用电话的用户又作了一次分类，1min的代表这些使用者占用电话时间小于等于1min，2min表示使用者占用电话时间小于等于2min，以此类推。根据电话使用规则，1min的用户只需要得到一次分配即可完成通话，而其他两类用户需要排队两次到三次。 电话的利用率 = sum (active use cpu time)/period 每一个分配到电话的使用者使用电话时间的总和去除以统计的时间段。这里需要注意的是是使用电话的时间总和(sum(active use cpu time))，这与占用时间的总和(sum(occupy cpu time))是有区别的。（例如一个用户得到了一分钟的使用权，在10秒钟内打了电话，然后去查询号码本花了20秒钟，再用剩下的30秒打了另一个电话，那么占用了电话1分钟，实际只是使用了40秒） 电话

的Average Load体现的是在某一统计时间段内，所有使用电话的人加上等待电话分配的人一个平均统计。 电话利用率的统计能够反映的是电话被使用的情况，当电话长期处于被使用而没有的到足够的时间休息间歇，那么对于电话硬件来说是一种超负荷的运作，需要调整使用频度。而电话Average Load却从另一个角度来展现对于电话使用状态的描述，Average Load越高说明对于电话资源的竞争越激烈，电话资源比较短缺。对于资源的申请和维护其实也是需要很大的成本，所以在这种高Average Load的情况下电话资源的长期“ 热竞争 ”也是对于硬件的一种损害。 低利用率的情况下是否会有高Load Average的情况产生呢？理解占有时间和使用时间就可以知道，当分配时间片以后，是否使用完全取决于使用者，因此完全可能出现低利用率高Load Average的情况。由此来看，仅仅从CPU的使用率来判断CPU是否处于一种超负荷的工作状态还是不够的，必须结合Load Average来全局的看CPU的使用情况和申请情况。 所以回过头来再看测试部对于Load Average的要求，在我们机器为8个CPU的情况下，控制在10 Load左右，也就是每一个CPU正在处理一个请求，同时还有2个在等待处理。看了看网上很多人的介绍一般来说Load简单的计算就是2* CPU个数减去1-2左右（这个只是网上看来的，未必是一个标准）。 补充几点： 1. 对于CPU利用率和CPU Load Average的结果来判断性能问题。首先低CPU利用率不表明CPU不是瓶颈，竞争CPU的队列长期保持较长也是CPU超负荷的一种表现。对于应用来说可能会去花时间在I/O,Socket等方面，那么可以考虑是否后这些硬件的速度影响了整体的效率。 这里最好的样板范例就是我在测试中发现的一个现象

：SIP当前在处理过程中，为了提高处理效率，将控制策略以及计数信息都放置在Memcached Cache里面，当我将Memcached Cache配置扩容一倍以后，CPU的利用率以及Load都有所下降，其实也就是在处理任务的过程中，等待Socket的返回对于CPU的竞争也产生了影响。 2．未来多CPU编程的重要性。现在服务器的CPU都是多CPU了，我们的服务器处理能力已经不再按照摩尔定律来发展。就我上面提到的电话亭场景来看，对于三种不同时间需求的用户来说，采用不同的分配顺序，我们可看到的Load Average就会有不同。假设我们统计Load的时间段为2分钟，如果将电话分配的顺序按照：1min的用户，2min的用户，3min的用户来分配，那么我们的Load Average将会最低，采用其他顺序将会有不同的结果。所以未来的多CPU编程可以更好的提高CPU的利用率，让程序跑的更快。 以上所提到的内容未必都是很准确或者正确，如果有任何的偏差也请大家指出，可以纠正一些不清楚的概念。 