文献综述

基于linux云计算平台上的进程迁移的研究

一、基本概念的概述

1.1 云计算

现在，行业对于云计算的定义并不一致，但一个比较一致的看法就是云计算是基于Web的一种服务，它消除了用户在传统硬件、软件、专业技能方面的投资，可以完全脱离技术与部署中的复杂性而轻松获取，它是由规模经济催生的大规模分布式计算范式，抽象的，虚拟化的，动态可扩展，管理计算能力，内存，平台和服务都是通过互联网按需提供给外部客户。

1.2 linux与云计算【19】

业界一致的观点就是云计算将架构在开源软件之上，并且大部分基础应用都将基于开源软件。因为大家都知道，作为集中式的服务平台，开放性永远是其关键要素之一，同时开源软件的灵活性和可扩展性也完全吻合云计算的发展趋势。

1.3 进程迁移的概念

进程迁移就是将一个进程从当前位置移动到指定的处理器上。其基本思想是在进程执行的过程中移动它，使得它在另一个节点上能够继续运行，继续运行时可以保留在源节点上已经执行的结果而不必重新执行该进程，用户丝毫感觉不到这种变化，所以对用户来说这意味着迁移是不中断的是透明的。

二、进程迁移应用的意义概述

国内外研究者通过对大量的负载平衡策略的研究表明进程迁移是实现负载平衡的基础，同时也是高容错性的一种非常有效的手段。如：

1、在某一时刻，有些节点负载较轻甚至空闲，而有些节点负载较重，此时将负载较重节点上的进程迁移到负载较轻或空闲的节点上来，让整个系统负载均衡来全面提高其性能，让资源得到充分的利用；

2、当某个节点出现故障时，将故障节点上的进程迁移到其它节点继续恢复运行，极大的提高了系统的可靠性和可用性；

3、传统的从文件服务器通过网络把数据传输给进程，我们也可以将进程迁移到文件服务器上进行 I/O。这样对于某些需要向文件服务器请求大量数据的进程，非常有效的减少了通讯量，提高了效率；

4、使用集群中某些机器的特殊能力。如果某个进程能够从集群中的某台特定机器上受益，它就应该在那台机器上执行。如进行数值计算的程序能够通过使用数学协处理器或超级计算机中的多个处理器:来缩短程序执行时

三、国内外对进程迁移的研究现状

由于进程迁移极大的改善了系统性能，国内外很多学者专家和同行们，对其进行了深入大量的研究，但是因为进程迁移的复杂性和其对OS的依赖性阻碍了进程迁移的广泛应用，当然也取得了很大的进展。现在有通过修改操作系统来进行进程迁移的，如：MOSIX【15】（MOSIX系统通过修改Linux内核的方法，实现了负载均衡的集群系统，MOSIX实现了透明的抢占式进程迁移机制，可以随时随地迁移几乎所有的进程）等；

也有在用户层实现进程迁移机制的，如：Condor【1】（Wisconsin-Madison 大学的Condor 系统，，Condor支持Unix工作站网络中的进程迁移。它实际是分配进程到空闲工作站的批处理程序， Condor进行进程迁移时将进程打包到一个文件进行传输，然后再恢复该进程）等；在这些系统中，大部分是基于检查点保存重起机制来实现的（尤其是Condor 利用检查点技术在工作站网络上实现了一个基于进程迁移的分布式作业管理系统，它的检查点技术是用户级检查点的典范，被许多容错系统使用或借鉴），都是这类算法属于全拷贝类型。采用此种算法使整个进程迁移过程有较大的时延。

Theimer【2】在V-system中提出了pre-copy（预拷贝）算法（当进程在源节点上运行时，并行的传输地址空间到目标节点上然后继续拷贝修改了的页面)但该算法将某些信息拷贝了两次增大了整个工作量。

Zayas【3】提出了请求页进程迁移算法（首先挂起进程只传送进程的状态，然后恢复进程的执行，地址空间的其他内容在新进程执行过程中出现页面错误时再通过请求页面，传输所需的页面内容）但该算法每页的传输代价很大，而且迁移完毕后对源主机的依赖性很大。

Sprite【10】系统支持在任意时间，进程在各主机之间移动，它采用的进程迁移机制不管是对被迁移进程还是用户都是高度的透明，其最大的问题是迁移后对源域有很大的依赖性，就是所谓的 “剩余依赖”。

ZAP【6】是为传统和网络应用提供了透明的进程迁移的系统，在操作系统上面提供一个虚拟层pods（process domains)，把进程对内核的接口如文件句柄，socket独立出来放到一个可以迁移的容器中，因为迁移单位变小所以迁移速度较快，但它不支持动态迁移，即不支持基于场景的迁移。

Collective project【7】把虚拟机的迁移当做一种方法，它可以让用户在不同时间使用不同的物理机，比如用户可以把操作系统的实例从家里迁移到工作单位的计算机上，此项目利用ASDL慢速连接，迁移时停止了OS的执行，减小映像文件的大小，使得总迁移时间较快。

NomadBIOS【8】系统是一个基于L4 微内核的虚拟化迁移系统，第一次引入了动态迁移的概念，即基于场景的迁移。它利用pre-copy的技术实现了尽可能小的停机时间，但没有解决wws（可以工作集）的问题。

国内的Wulor-75/32系统【11】，研究人员首次在上面成功的进行了进程迁移，它在每个处理机上设置一个“服务器进程”和“顾客进程”，顾客进程负责调度作业并迁移外来进程，服务器进程则负责执行顾客和其他服务器所要求的服务，让核心通过顾客进程陷入，此时要迁移进程已经被切换，而与其相关的环境被保护到U区，这样在目的节点上，只需将这些环境恢复到新创建进程的U区中，当新进程被调度时，环境就会自动恢复。

四、进程迁移算法和主要的实现步骤

4.1 以现有的分布式进程迁移算法为参考，主要有以下几种进程迁移算法：【18】

1、贪婪拷贝算法：该算法先挂起源主机进程，然后传送进程的全部状态（包括一些打开的文件和执行状态等）到目标主机后，再启动日标主机进程。这种算法简单．易于实现，但存在两点不足：延时较长，这对于实时系统是不可接受的；传输冗余数据，而实际并没有用上，造成网络负担。

2、惰性拷贝算法：先传输进程在目标主机上重新执行所需要的最小相关信息。比起贪婪拷贝算法，它传输的足必需的最少量的状态集合，然后在主机上启动。这些信息通常是进程的部分或全部核心数据和一小部分（二三页）地址空间。当进程在目标主机上执行需要其余状态信息时，再传输这些信息。其优点是延迟小、网络负担少，缺点是会导致对源主机的剩余依赖性，因此不能提高系统的可靠性。

3、预拷贝算法：与前面两种算法不同的是．预拷贝算法在进程的部分或全部地址空间从源

主机到目标主机传输完毕时，源主机才挂起进程并传输核心数据。也就是说．当进程在源主机上执行时，并行传输地址空间到目标主机上：进程挂起后再传输核心数据(包括打开的文件、执行状态、当前目录等)，一些先前已经传输而后被改变的地址空间一起传输到目标主机上。这样就会产生一个问题：这种算法虽然降低了进程挂起的时间，避免困挂起时问长而导致的开销和错误，但是却会将某些信息拷贝两次，总的传输时间反而增长。

4、Demand—Page算法：一开始只传送进程的执行状态、通信状态，进程的地址空间保留在源节点上，然后恢复进程的运行，目的节点仅在需要访问某些页时才请求之。

5、file-server 算法：引入了第三台机器——文件服务器，它和Demand—page算法类似，一开始也传送除进程地址空间之外的进程状态信息给目的节点，然后将进程地址空间所涉及的内存页传送到文件服务器，之后恢复进程的运行，目的节点发生缺页时向文件服务器申请相关页。

4.2 进程迁移的步骤

虽然有很多不同的迁移实现和设计，但大部分可归结为以下几个步骤：【4】

1、一个迁移请求发出到远程节点。经过谈判，迁移已被接受；

2、一个进程通过暂停其执行脱离其源节点，宣称它是在迁移状态，并按照下面描述的第一步暂时重定向其通信；

3、通过排队到达的消息定向到已经迁移的进程来暂时重定向通信，然后在迁移之后将它们提供给进程，只要有额外传入的消息，这一步就和后面的4,5,6并行进行，一旦进程迁移后启用通信渠道，迁移的进程就可以被外界所知。

4、进程状态的提取包括内存的内容，处理器的状态（寄存器的内容），通信状态（例如打开的文件和通信的通道）以及与内核相关的上下文环境。通信的状态和内核的上下文环境都是依赖于操作系统的，一些本地操作系统的内部状态是不被转移的，进程的状态通常一直保留在源节点上直到迁移结束，甚至在某些系统上一直保留到迁移完全完成。处理器的依赖，比如栈和寄存器的内容在异构迁移时不得不被淘汰。

5、目标进程实例的创建和进程信息的导入：如果没有从源进程实例中传输足够的进程信息，那么目标进程不会创建和激活。如果完成足够的进程信息传输，则目标进程实例将被提升为常规进程。

6、状态在远程节点上被转移成一个新的实例，不是所有的状态都需要被转移，一些状态可能在迁移完成后才被需要。

7、必须维护被迁移进程的一些转发引用方式。这对于进程通信和进程控制是需要的。有几种方式可以实现：在源节点上注册当前位置（Sprite 使用的方式）、搜索被迁移的进程（V Kernel 在通信协议中实现）、在所有可以访问的节点中转发消息。本步骤也启动被迁移进程的通信通道。因为通信通道已经持久的重定向，第三步的队列化可以终止。

8、当足够的状态被转移时，新的实例进程即可被恢复，有了这一步，迁移的过程就已经完成了，一旦所有的状态都从源节点上被转移后，那么源节点就有可能被删除了。

五、设计进程迁移功能要解决的关键问题

1、谁来启动迁移过程?

这取决于迁移机制的目标。如果目标是负载平衡，那么，通常由操作系统中掌管系统负载的组件决定什么时候进行迁移，若其目的在于获得特定资源，那么，可由需要资源的进程自行决定何时进行迁移，这种迁移也称为自迁移（Self-migration）。

2、要迁移进程的什么部分?

一个进程的状态包括内存空间（大量的状态都和内存的状态有关，如代码和数据）；打开文件（包括打开文件的内部标识，打开文件的位置和文件缓冲块）；进程消息（如果OS是基于

消息的，其状态包括收和发的缓冲消息）；执行状态（包括在进行上下文切换时有关的核心存储和恢复信息，如寄存器的值）；其他内核信息（OS存储进程的其他信息，如当前的工作目录和进程ID）。程序执行通常涉及到跟踪过程调用和过程间参数传递的栈；此外还有与每个进程相关联的，用于控制进程的其它操作系统。属性的集合通常被称为PCB（进程控制块），PCB中有：标志符、用户可见寄存器、控制和状态寄存器、栈指针、调度和状态信息、进程间通信信息、存储管理数据结构等。通常把程序、数据、栈和属性的集合称作进程映像。要保证迁移后的进程能够正确的继续进行，必须要迁移进程映像。从执行的角度看，进程控制块移动的困难之处在于进程地址空间的传送和进程打开文件的传送。进程地址空间的传送，假设用虚拟存储策略(分段或分页)，有两种方法： 要么迁移时传送整个地址空间或者仅传送那些在主存中的地址空间部分，在需要时再传送虚拟地址空间中的段。这样的话 ，处于运行状态的进程就依赖于原节点上的进程信息，这种状态称为“残余依赖”。

3、未完成的消息和信号如何处理?一般在集群系统中，为了进行进程迁移需要再进行以下的修改：① 必须对文件系统进行一定的修改，使每个机器看到相同的名字空间；②必须传送足够的状态，从而确保正常的核心调用能够在远端机器上执行；③ 一些特殊的核心系统调用如gettimeofday、getpgrp应该发回到原始节点执行

这个问题是关于消息和信号在迁移过程中的处理，可通过一种机制进行处理：在迁移进行时，暂时存储那些完成的消息和信号，然后将它们直接送到新的目的地，有必要在迁移出发的位置将正在发出的信息维持一段时间，以确保所有的未完成消息和信号都被传送到目的地。

总结：

本文总结了进程迁移的实践意义以及实现进程迁移的主要算法和主要的步骤，分析了当前进程迁移在国内外的研究现状。由于进程迁移的复杂性以及对操作系统的依赖性，使得进程迁移没有得到广泛的应用。以前的研究在选择究竟是哪个进程适合迁移，应该迁移到哪个节点上都是考虑某一方面的因素，选择一种特定的已有的迁移算法，并非最优，目前我的工作是在linux云平台上考虑进程的综合因素（时间，空间，权值等）进行迁移，并模拟实验环境，不断的验证和分析自己提出的算法的优良性。

六、参考文献

[1] Checkpoint and migration of UNIX processes in the Condor distributed processing system M Litzkow, T Tannenbaum, J Basney... 1997

[2] Marvin M. Theimer, Keith A. Lantz, and David R. Cheriton Preemptable Remote Execution Facilitiesfor the V-System 2002

[3] Edward R. Zayas Attacking the Process Migration Bottleneck

[4] DEJAN S. MILO´JI˘CI´C *HP Labs* FRED DOUGLIS *AT&T Labs–Research* YVES PAINDAVEINE *TOG Research Institute* RICHARD WHEELER *EMC* AND SONGNIAN ZHOU *University of Toronto and Pla* Process Migration ACM Computing Surveys (CSUR) Surveys Homepage archive Volume 32 Issue 3, Sept 2000

[5] Performance of PVM with the MOSIX Preemptive Process Migration Scheme *

[6] Steven Osman, Dinesh Subhraveti, Gong Su, and Jason Nieh The design and implementation of Zap: a system for migrating computing environments ACM SIGOPS Operating Systems Review - OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation 2002

[7] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen†, Eric Jul†, Christian Limpach, Ian Pratt, Andrew Warfield Live Migration of

Virtual Machines NSDI'05 Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2 2005
[8] MASTER'S THESIS:NOMADIC OPERATING SYSTEMS
[9] *Jonathan M. Smith* A Survey of Process Migration Mechanisms ACM SIGOPS Operating Systems Review 1998
[10] FRED DOUGLIS* AND JOHN OUSTERHOUT Transparent Process Migration: Design Alternatives and the Sprite Implementation SOFTWARE—PRACTICE AND EXPERIENCE, VOL. 21(8), 757–785 1991
[11] 肖红，邱毓兰，彭德纯 分布式计算系统中进程迁移的方法 软件学报 1994
[12] 周全，卢显良，任立勇，刘晓燕 基于Linux的进程迁移机制设计 计算机应用 2003
[13] 刘天田 杨升春 欧中红 袁由光 基于消息传递并行进程迁移技术的研究与实现 计算机科学 2009
[14] 张福新， 章隆兵 UNIX 系统用户级检查点技术分析 第八届学术研讨会论文 2004
[15] 黄翊，蒋江，张民选 MOSIX进程迁移机制研究 《计算机工程》 -2002年8期
[16] 李永 进程迁移技术分析和研究 《中国科技博览》 -2010年7期
[17] 庞骏林，蒋翠玲 进程迁移研究 计算机工程与科学 2001年第23卷第5期
[18] www.baidu.com
[19] 土木 云计算 linux 2009