龙芯中科
LOONGSON TECHNOLOGY

中间件在多路平台上的 NUMA 调优指南

北京市海淀区中关村环保科技示范园龙芯产业园2号楼
Building No. 2, Loongson Industrial Park, Zhongguancun Environmental Protection Park, Haidian District Beijing 100095, P. R. China
www.loongson.cn

文档更新记录	文档编号:	
	文档名:	中间件在多路平台上的 NUMA 调优指南
	版本号	0.5
	创建人:	王雪倩
	创建日期 :	2016.10.11

更新历史

序号.	更新日期	更新人	更新内容
1	2016.9.18	王雪倩	第一版
2	2016.10.11	王雪倩	添加附录 A：3B1500、3B2000 四路服务器启动脚本；中间件需要修改的端口号中添加 <listener>
3			
4			
5			

目录

1. NUMA 调优原理

在龙芯的多路服务器上，可能会出现这样一种现象：多路服务器的性能比单路服务器的性能提升不明显。典型的是运行中间件这种多线程的并发应用。

原因是多路服务器上是有多个 CPU 和多个物理内存条，普遍使用的 SMP（对称多处理）模型，即操作系统将多个物理内存条作为一大块连续内存，所有 CPU 都能够访问到这个共享内存的所有单元。但是，由于处理器之间互联进行通讯时要有一定的开销，因此所有 CPU 到所有物理内存的访问速度是不同的，是 NUMA（Non-Uniform Memory Access）模型。

以图 1 为例，一台机器有 2 个处理器（cpu0, cpu1），在每个处理器上接两个内存条（memory0.1, memory0.2, memory1.1,memory1.2），总共有 4 个内存块。一个处理器和直接相连的两个内存条合起来，称为一个 NUMA node，这样这个机器就会有两个 NUMA node。在物理分布上，一个 NUMA node 之内的处理器和内存块的物理距离更小，因此访问也更快。而如果跨 node 访问内存，则速度会变慢。在 NUMA node1 中，cpu1 访问 memory1.1 和 memory1.2 就比访问 memory0.1 和 memory0.2 更快。

基于这一个原理，可以解释“多路服务器的性能比单路服务器的性能提升不明显”的问题。对于运行在多路服务器上的多线程应用程序，操作系统默认会随机分配物理内存，从而导致线程运行在 CPU0 上、而访问的数据位于 CPU1 所连接的 memory 上，这样会导致应用程序运行速度下降。

为了充分发挥多路服务器上应用程序的性能，可以利用操作系统提供的命令工具，对应用程序使用的 CPU 和物理内存进行绑定，保证本 node 内的 CPU 只访问本 node 内的内存块。本文档就介绍这样一种优化方法，在实施之后一般都能够使多路服务器上的中间件提升 80%性能。

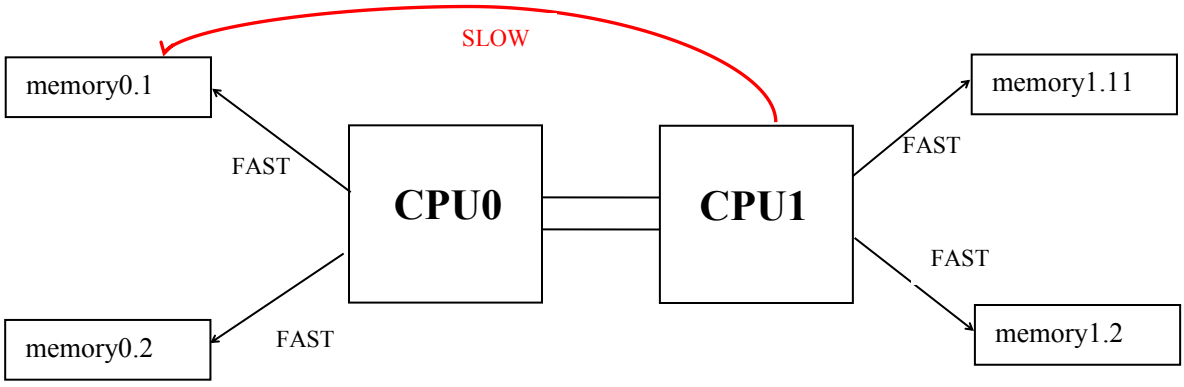

图 1 NUMA 架构

2. 多路服务器的 NUMA 绑定优化方法

比如一个 OA 系统在 3B2000 双路服务器上运行，进行压力测试时发现性能较低，不能满足用户需求，那么接下来就需要进行多路服务器的 NUMA 绑定调优。以 3B2000 双路服务器为例，介绍具体实现方法。

2.1 查看系统 cpubind 数

在终端中执行命令（以 3B2000 双路服务器为例）：

```
# numactl --show

policy: default

preferred node: current

physcpubind: 0 1 2 3 4 5 6 7

cpubind: 0 1

nodebind: 0 1

membind: 0 1
```

cpubind 后面的数字说明机器是 2 路，意味着需要绑定 2 个中间件。

如果是其他类型服务器，详情请参见**附录 A**。

2.2 复制多份中间件，修改端口号

首先要在一台服务器上复制 2 份中间件，每一个中间件是一个独立的目录。

为了能够在一台机器上启动两个中间件进程，两个中间件必须使用不同的网络端口号，因此需要修改第二份中间件的配置文件。端口号修改以“同一个 IP 地址上的端口号不冲突”为原则,具体端口号可以任意指定，但是需要在本机保证唯一、并且和第一份中间件相区别。

以东方通中间件为例，配置文件中需要修改的端口号有 3 个（http、jmx 和管理控制台端口）,具体需要修改的是 config 目录下的 twns.xml 文件：

将<http-listener>端口号由 8080 改为 8081；

将<jmx-service>端口号由 7200 改为 7205；

将<admin-listener>端口号由 9060 改为 9061；

将<listener>端口号由 5100 改为 5101；

2.3 对中间件启动脚本进行 numa 绑定

在两份中间件的 bin/目录下，分别编写 numa 绑定的启动脚本。脚本中调用操作系统内置的 numactl 命令工具，加上适当参数，使每一个中间件只运行在一个 CPU 上，只访问该 CPU 节点内部的物理内存。

numactl 命令在中标操作系统服务器版中默认安装。

第一份中间件启动脚本：

```
# numactl --cpunodebind=0 --membind=0 ./startserver.sh
```

第二份中间件启动脚本：

```
# numactl --cpunodebind=1 --membind=1 ./startserver.sh
```

分别执行上面两个脚本，把两个中间件都启动起来。

2.4 配置负载均衡器

使用 numa 绑定策略进行调优，在一台服务器上会启动多个中间件。为了保证 OA 系统对用户有唯一的入口 URL，需要应用端有一个 Web 负载均衡器，做为多个中间件的调度入口。

Web 负载均衡器可以是软件负载均衡或者是硬件负载均衡，软件负载均衡是使用操作系统中内置的 Apache 服务器实现，硬件负载均衡通过一台额外的硬件设备来实现。推荐使用硬件负载均衡器，在管理上更加方便，性能也更高。

Web 负载均衡设备的实现原理是在前端做为用户访问的总入口，把外界 Web 访问请求转发到后端的多台 Web 服务器（即中间件）上，这个过程对用户端是透明的。用户实际上不知道服务器是做了负载均衡的，因为他们访问的还是一个目的 URL。

对于系统的管理员，需要在负载均衡器中进行配置，把一台多路服务器上启动的两个中间件的 URL 都添加负载均衡器中。由于在不同厂家的负载均衡器中，具体的配置方法会有区别，请参见负载均衡器的文档。

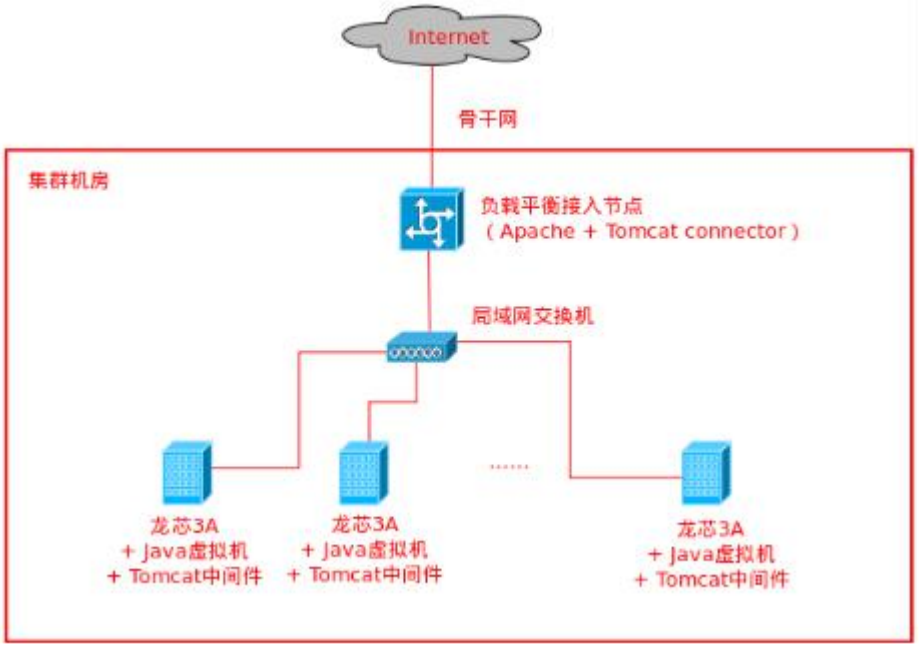

图 2 Web 负载均衡器

2.5 检查 numa 绑定状态

上面的步骤都完成后，现面可以进行性能的测试。在测试过程中，可以利用以下方法检查绑定状态是否正常：

1）使用 top 命令：在命令行上运行 top 命令，按“1”键，可以查看每个 CPU 的使用情况。当启动第一个中间件后，会发现只有前 4 个 CPU 核（0-3）有负载，尤其是有压力的情况下，负载明显上升到 80%以上，而另外 4 个 CPU 核（4-7）负载几乎为 0；继续启动第二个中间件后，则后 4 个 CPU 核上才会出现负载。

2）使用 ps 命令：在命令行上运行 ps ax 命令，可以查看看中间件的进程号 PID。在操作系统的/proc/[PID]/numa_maps 文件中，可以查看到 numa 状态，尤其是所使用的每一块物理内存的位置。这种方法对技术水平要求较高，详细说明请自行查找资料。

3. 性能提升案例

在龙芯 3B2000 双路服务器上，使用 LoadRunner 对某 OA 系统进行压力测试，优化后的性能有明显提升：TPS 从 150 提升到 310，提升近一倍；响应时间从 3s 减少到 1.3s，减少近一半；应用服务器的 CPU 占用率下降 20%左右。

证明本文的方法能够有效提升多路服务器上中间件的性能。

附录 A

3B1500 服务器：

```
# numactl --show

policy: default

preferred node: current

physcpubind: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

cpubind: 0 1 2 3

nodebind: 0 1 2 3

membind: 0 1 2 3
```

cpubind 后面的数字说明机器是 4 路，意味着需要绑定 4 个中间件：

1. 将中间件复制成 4 份，分别修改其他 3 份的配置文件端口号，以互相不冲突为标准（见 2.2 复制多份中间件，修改端口号）；

2. 中间件启动脚本：

```
第一份中间件：

# numactl --cpunodebind=0 --membind=0 ./startserver.sh

第二份中间件：

# numactl --cpunodebind=1 --membind=1 ./startserver.sh

第三份中间件：

# numactl --cpunodebind=2 --membind=2 ./startserver.sh

第四份中间件：

# numactl --cpunodebind=3 --membind=3 ./startserver.sh
```

3B2000 四路服务器：

```
# numactl --show

policy: default

preferred node: current

physcpubind: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

cpubind: 0 1 2 3

nodebind: 0 1 2 3

membind: 0 1 2 3
```

cpubind 后面的数字说明机器是 4 路，意味着需要绑定 4 个中间件：

1. 将中间件复制成 4 份，分别修改其他 3 份的配置文件端口号，以互相不冲突为标准（见 2.2 复制多份中间件，修改端口号）；

2. 中间件启动脚本：

```
第一份中间件：

# numactl --cpunodebind=0 --membind=0 ./startserver.sh

第二份中间件：

# numactl --cpunodebind=1 --membind=1 ./startserver.sh

第三份中间件：

# numactl --cpunodebind=2 --membind=2 ./startserver.sh

第四份中间件：

# numactl --cpunodebind=3 --membind=3 ./startserver.sh
```