实现复合应用程序的高可用性

http://www.linuxeden.com/html/sysadmin/20090224/64224.html

本文描述使用 Linux-HA 为复合应用程序实现高可用性。向复合应用程序交付高可用性具有很大的挑战性。由于复合应用程序由一些不同类型的应用程序组成，每个应用程序都具有不同的可用性需求，所以配置相当复杂。在本文中，作者描述如何为复合应用程序 Tivoli® Maximo® 设计和实现一种高可用性原型。其中的配置脚本展示了如何使用系统化和优先化的故障转移计划，向由相关应用程序组成的异构集群提供高可用性。

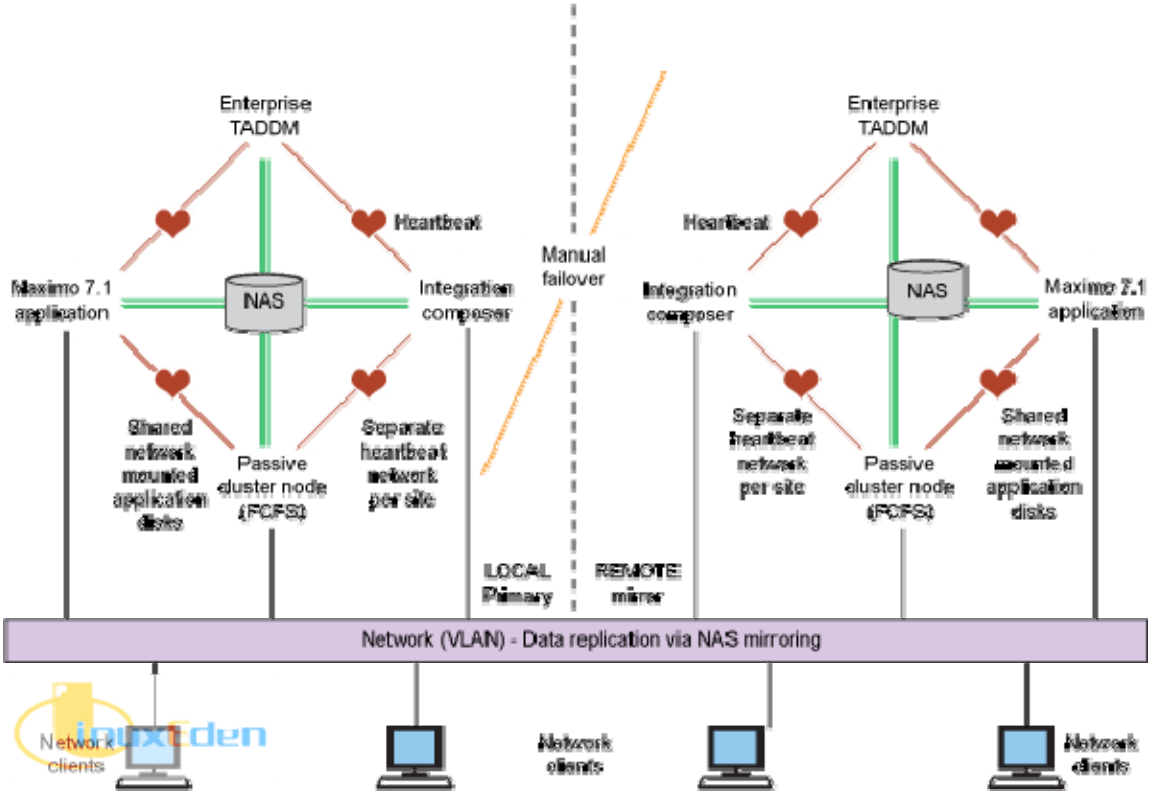
在 2008 年，我们为一家大型在线企业开发了针对我们的 CMDB 实现（配置管理数据库）Tivoli Maximo and Tivoli Application Dependency Discovery Manager 的高可用性解决方案。这家企业现在采用了 Maximo 的一种成熟的 CCMDB 实现（现在称为 Tivoli Service Request Manager and Tivoli Asset Manager for IT），其中包含一个单一的企业级 Maximo（CCMDB 版本）。我们的目标在于使用多个域 CMDB 应用程序从不同的企业站点收集信息，并将这些信息聚合到一个企业 CMDB 实例中。对这些企业 CMDB 数据进行过滤并累积到 CCMDB 实例 Maximo 中。

Maximo 既存储预设策略所描述的企业 WISB 的理想 CMDB 状态（“应该处于的状态”），还 存储 CI 信息所表示的 WIRI 的实际状态（实际处于的状态），CI 信息收集自企业中部署的成千上万个服务器和应用程序。

一般而言，HA 层次结构中的不同功能节点要求不同的 HA 设计。例如，网关服务器、普通 Windows® 计算机要求 MSCS。域 CMDB 应用程序要求 HA，但是使用冷备份就足够了。而 Maximo 需要 24/7 可用性，并且 Maximo 和企业 CMDB 连接到自己的专用数据库 — 这些数据库通常是集群的一部分。但是在本文的原型中，我们主要讨论应用程序的 HA 可用性，因此我们仅针对应用程序可用性进行设计。图 1 给出了我们的设计。

图 1. 针对 Maximo and Enterprise Tivoli Application Dependency Discovery

Manager 的 HA 灾难恢复（DR）：两个 4 节点 HA 集群



在本文中，我们描述一种管理高可用性、异构、包含多个应用程序的节点集群的方法。在高可用性集群中，每个应用程序都具有不同的可用性配置文件，这使我们的解决方案实际上比面向单一应用程序（比如数据库）的简单高可用性解决方案更加复杂。

我们的解决方案将采用一种算法（协议）来适应高可用性集群中应用程序（或节点）的第一次和 第二次故障。这个协议为集群中的每个应用程序节点提供了精确的故障转移顺序。 它考虑了这样的情形：性能不能降低，而且由于运行时冲突，不能在同一计算机上运行多个应用程序。

设计和实现特性包括：

1. 尽管集群中包含 3 个应用程序（Enterprise CMDB、ITIC 和 Maximo），但 Maximo 拥有最高的优先级。因此，无论哪个应用程序节点失败，Maximo 都一定能被客户机使用（客户机将调用 Maximo API）。
2. 我们使用不均衡的节点权重来部署自动恢复功能，这样当一个宕机的服务器恢复之后，应用程序将恢复到该服务器中。
3. 我们的设计规范要求管理集群中的两次故障，但是，即使是在出现了 3 次故障之后，我们仍将能够设计和实现完整的 CCMDB 可用性（Maximo）。
4. 我们在最初的设计中未考虑 quorum 服务器，但是我们不得不使用它。
5. 将 Linux®-HA 配置为按照这种针对 1、2、3 次故障的严格故障转移规则集操作具有一定的挑战性。
6. 我们在灾难恢复（DR）站点（图 1 中的远程镜像站点）重复这种设计。所有站点中的故障转移分为自动和手动两种：自动 HA、手动 DR。

7. 为了优化机器的使用并维护 HA，我们在每个站点设计了一个 4 节点集群：一个节点对应一个应用程序，剩下的一个节点充当故障转移节点。

HA 架构通常用于单一软件类型，比如数据库或 Web 服务器。在我们的示例中，我们将展示如何实现 CCMDB 等应用程序的高可用性，该实现包含 3 个独立的软件组件：

- Tivoli Application Dependency Discovery Manager (TADDM)：通过发现应用程序依赖关系和配置，为 IT 服务管理提供可见性。
- IBM Tivoli Integration Composer (ITIC)：支持 Tivoli Asset Management for IT 与资产目录和系统管理工具的快速集成。
- Maximo：在一个统一平台上提供对所有资产类型的全面的资产生命周期和维护管理。

安装 HA

Linux-HA 安装是一个简单、直观的过程（参见 参考资料，获取软件）。确保系统拥有正确的补丁级别，以满足心跳（heartbeat）软件的前提条件。我们用于演示的 Linux-HA 的版本为 2.1.4。

当安装完成时，重启计算机。必须执行这一步。为组成集群的所有 4 台计算机执行清单 1 中的步骤。

清单 1. 安装 HA

```
[root@hacluster2 tmp]# rpm -ivh
perl-TimeDate-1.16-3_2.0.el5.noarch.rpm
warning: perl-TimeDate-1.16-3_2.0.el5.noarch.rpm: Header V3 DSA
signature:
 NOKEY, key ID 66534c2b
Preparing ...
########################################### [100%]
   1:perl-TimeDate
########################################### [100%]
[root@hacluster2 tmp]# rpm -ivh
heartbeat-pils-2.1.4-2.1.i386.rpm
warning: heartbeat-pils-2.1.4-2.1.i386.rpm: Header V3 DSA
signature:
 NOKEY, key ID 1d326aeb
Preparing ...
########################################### [100%]
   1:heartbeat-pils
########################################### [100%]
[root@hacluster2 tmp]# rpm -ivh
heartbeat-stonith-2.1.4-2.1.i386.rpm
warning: heartbeat-stonith-2.1.4-2.1.i386.rpm: Header V3 DSA
```

```
signature:
 NOKEY, key ID 1d326aeb
Preparing ...
########################################### [100%]
   1:heartbeat-stonith
########################################### [100%]
[root@hacluster2 tmp]# rpm -ivh heartbeat-2.1.4-2.1.i386.rpm
warning: heartbeat-2.1.4-2.1.i386.rpm: Header V3 DSA signature:
NOKEY, key ID 1d326aeb
Preparing ...
########################################### [100%]
   1:heartbeat
########################################### [100%]
[root@hacluster2 tmp]#
```

配置 HA

下一步是创建 ha.cf 文件。 创建以下文件：/etc/ha.d/ha.cf。ha.cf 文件存储关于此设置涉及到哪些节点的信息。

在分配为 DC（designated coordinator）的机器上创建 ha.cf。创建此文件和 authkeys 文件之后，使用 FTP 或简单 SCP 将它们复制到其他机器。因为 4 个节点（3 个应用程序和一个备用节点）都位于此集群中，我们的 ha.cf 文件类似于清单 2：

清单 2. HA 配置文件 ha.cf

```
node hacluster1.svl.ibm.com hacluster2.svl.ibm.com
hacluster3.svl.ibm.com
     hacluster4.svl.ibm.com
bcast eth0
crm on
```

在清单 2 中：

- node 是一个指令，用于列出此集群中包含的节点。
- bcast 指令表示节点将通过此接口彼此通信并执行 ping 操作。
- crm 指令指定 heartbeat 是否应该运行一个支持两个或更多节点的集群管理器。

在清单 2 中，我们键入属于此集群的机器的主机名。接下来，我们需要向 /etc/ha.d/authkeys 添加一个验证密钥。清单 3 给出了我们使用的示例：

清单 3. 示例验证文件 authkeys

```
#
#       Authentication file. Must be mode 600
#
#
#       Must have exactly one auth directive at the front.
#       auth    sne authentication using this method-id
#
#       Then, list the method and key that go with that method-id
#
#       Available methods: crc sha1, md5. Crc doesn't need/want
a key.
#
#       You normally only have one authentication method-id
listed in this file
#
#       Put more than one to make a smooth transition when changing
wuth
#       methods and/or keys.
#
#
#       sha1 is believed to be the "best", md5 next best.
#
#       crc adds no security, except from packet corruption.
#               Use only on physically secure networks.
#
auth 1
1 sha1 haclusteringisfun
```

注意：这个文件的权限必须为 0600。当这些文件创建之后，通过发出命令 /etc/init.d/heartbeat start 启动 heartbeat。

在集群中的所有机器上运行清单 4 中所示的命令：

清单 4. 启动高可用性服务

```
 [root@hacluster1 heartbeat]# /etc/init.d/heartbeat start
Starting High-Availability services:

[   OK   ]
```

```
[root@hacluster1 heartbeat]#
```

如果看到这条消息，则意味着您的 Linux-HA 安装成功完成。现在，是时候测试复合应用程序的高可用性了。

添加一个 quorum 服务器

在一个 2 节点集群中，当一个节点发生故障或网络连接中断，每个节点都会充当主要节点并开始与外界交互。这是一种不利的竞态条件。我们需要一个外部仲裁程序来要求其中一个节点暂时停止活动。

如果其中一个机器崩溃，仲裁程序将让该机器暂时停止活动。这个仲裁程序称为 quorum 服务器，它可以是集群中的每个节点都可以到达的任何机器。将这个仲裁程序命名为 quorum 服务器是因为它运行一个 quorum 监控程序。修改 ha.cf，将下面这行代码添加到每个 ha.cf 文件：

清单 5. 在 ha.cf 中标识 quorum 服务器

```
cluster ourcmdb
quorum_server hacluster4.svl.ibm.com
```

quorum 服务器机器不一定要运行 heartbeat，但是我们建议安装 heartbeat，以便能够访问安装 heartbeat 时创建的所有二进制文件和目录路径（比如 /etc/ha.d）。在 quorum 服务器机器上，编辑文件 /etc/ha.d/quorumd.conf 并添加以下代码：

清单 6. 配置 quorum 服务器

```
cluster ourcmdb
version 2_1_4
interval 1000
timeout 5000
takeover 3000
giveup 2000
```

然后，使用 quorumd 启动 quorum 监控程序。确保它在计算机每次重启时都会启动。要自动启动 quoromd，将其添加到 inetd 中。

Linux-HA 使用一个名为 cib.xml 的配置文件，在集群中的所有节点上启动 heartbeat 时会自动创建该文件。这个 cib.xml 文件存储应用程序配置（指定哪个应用程序拥有更高的优先级，包括针对高可用性的规则）。您可以使用 GUI 工具（/usr/bin/hb_gui）修改 cib.xml，这是推荐方法，也可以手动修改。

Cib.xml 包含以下信息：

- 配置信息：
  - 集群节点信息
  - 资源信息
  - 资源限制
- 状态信息：
  - 哪些节点启动/关闭
  - 节点属性
  - 哪些资源在何处运行

因为 cib.xml 受到 heartbeat 进程的控制，所以应避免在集群运行时修改此文件。

注意：cib.xml 上的权限必须为 0600，并且必须归 haclient:hacluster 所有。

Linux-HA 附带了一组基于开放集群框架（Open Cluster Framework，OCF）的资源代理，该框架是用于实现高可用性的标准。在我们的场景中，因为所有应用程序都是定制的，我们必须为复合应用程序中包含的各个软件组件构建 OCF 资源代理。

配置 heartbeat

清单 7 给出了我们使用的 heartbeat 版本中的资源配置。配置文件位于 /var/lib/heartbeat/crm/cib.xml。基本而言，此文件指定集群的资源以及应该在何处执行这些资源。

我们为此场景开发了一个 cib.xml 文件，参见 参考资料 获取链接。我们为其添加了注释，以使该进程更容易执行。

清单 7. heartbeat 版本 2 中的资源配置

```
<cib generated="true" admin_epoch="0" have_quorum="true" ignore_dtd="false"
num_peers="2"
 ccm_transition="2" cib_feature_revision="2.0" crm_feature_set="2.0"
epoch="3" dc_uuid="
 ad893965-d27d-4908-a2ea-868f1661f644" num_updates="3" cib-last-written="Fri
Nov  14
 10:14:40 2008">
  <configuration>
    <crm_config>
      <cluster_property_set id="cib-bootstrap-options">
        <attributes/>
      </cluster_property_set>
    </crm_config>
    <nodes> /* Names of the nodes in the cluster */
```

```
        <node id="ad893965-d27d-4908-a2ea-868f1661f644"
         uname="hacluster1.svl.ibm.com" type="normal"/> /* Maximo node */
        <node id="5994eb92-0a13-4fc7-ab41-76098672fdbb"
         uname="hacluster3.svl.ibm.com" type="normal"/> /* ITIC node */
        <node id="c14b9082-5b1a-481c-930a-561e926df7c3"
         uname="hacluster4.svl.ibm.com" type="normal"/> /* eCMDB node */
        <node id="827b7884-06db-4d8d-994d-7e743b9bb969"
         uname="hacluster2.svl.ibm.com" type="normal"/> /* Spare node */
      </nodes>
      <resources>
        <group id="maximo_group">
         /* Assign base parameters for each application - Maximo */
          <primitive class="lsb" id="maximo_id" type="maximo">
            <operations>
              <op id="1" name="monitor" interval="10s"/>
              <op id="2" name="start" start_delay="10s"/>
            </operations>
            <meta_attributes id="063383a7-2c60-4cf0-b3b0-a3670328c3b8">
              <attributes> /* Priority are set as weighting factors. This
determines which
                           application will be placed on the spare node in
the event of
                           a second failure. Here, Maximo > eCMDB > ITIC.
Note that the
                           first failure pushes the downed application onto
the spare.
                           A subsequent failure determines whether the
initially downed
                           application or the newly downed application takes
precedence.
                         */
                <nvpair name="priority" value="3"
                 id="f0c58dd3-43bb-4a1e-86cc-8993e58ba399"/>
              </attributes>
            </meta_attributes>
          </primitive>
        </group>
        <group id="iticd_group"> /* Assign base parameters for each application
- ITIC */
          <primitive class="lsb" id="itic_id" type="iticd">
            <operations>
              <op id="3" name="monitor" interval="10s"/>
              <op id="4" name="start" start_delay="10s"/>
            </operations>
```

```
          <meta_attributes id="cb35cbb3-3241-4bf6-9ab7-f06d6f5baf89">
            <attributes>
              <nvpair name="priority" value="2"
               id="7f7a35eb-aff8-4f7d-8e4d-ec6a0414d6da"/>
            </attributes>
          </meta_attributes>
        </primitive>
      </group>
      <group id="taddm_group"> /* Assign base parameters for each application
- eCMDB */
        <primitive class="lsb" id="taddm_id" type="taddm">
          <operations>
            <op id="5" name="monitor" interval="10s"/>
            <op id="6" name="start" start_delay="10s"/>
          </operations>
          <meta_attributes id="e7baab9a-1460-423f-aabd-f68137d00d42">
            <attributes>
              <nvpair name="priority" value="1"
               id="0448d632-f8e7-4347-90bd-de8222b77bda"/>
            </attributes>
          </meta_attributes>
        </primitive>
      </group>
    </resources>
    <constraints>
      <rsc_colocation id="not_same_1" to="maximo_group" from="iticd_group"
       score="-INFINITY" symmetrical="false"/> /* Maximo application should
not run on
                                                the node assigned to ITIC
*/
      <rsc_colocation id="not_same_3" to="maximo_group" from="taddm_group"
       score="-INFINITY" symmetrical="false"/> /* Maximo application should
not run on
                                                the node assigned eCMDB */
      <rsc_colocation id="not_same_2" to="taddm_group" from="iticd_group"
       score="-INFINITY" symmetrical="false"/> /* eCMDB application should
not run on
                                                the node assigned to ITIC
*/
      <rsc_location id="location_maximo" rsc="maximo_group">
        <rule id="prefered_location_maximo_1" score="20">
          <expression attribute="#uname" operation="eq"
value="hacluster2.svl.ibm.com"
           id="a4b1be4e-4c25-46a6-9237-60a3b7b44389"/> /* Spare node for
```

```
Maximo should
                                                                preferred node
fail */
         </rule>
         <rule id="prefered_location_maximo_2" score="100">
           <expression attribute="#uname" operation="eq"
value="hacluster1.svl.ibm.com"
            id="3ece30c7-0530-4b54-a21b-ace9b127d3e3"/> /* Preferred node for
Maximo
                                                                to run */
         </rule>
         <rule id="prefered_location_maximo_3" score="-INFINITY">
           <expression attribute="#uname" operation="eq"
value="hacluster4.svl.ibm.com"
            id="7cb3b3e2-191f-492d-84f9-257b96c02c3c"/> /* Maximo cannot
co-exist with
                                                                eCMDB on this
node */
         </rule>
         <rule id="prefered_location_maximo_4" score="-INFINITY">
           <expression attribute="#uname" operation="eq"
value="hacluster3.svl.ibm.com"
            id="a78cce70-c717-4c33-910e-11cc39ded186"/> /* Maximo cannot
co-exist with
                                                                ITIC on this node
*/
         </rule>
       </rsc_location>
       <rsc_location id="location_iticd" rsc="iticd_group">
         <rule id="prefered_location_iticd_1" score="20">
           <expression attribute="#uname" operation="eq"
value="hacluster2.svl.ibm.com"
            id="e3de5eee-ee29-4e94-89d7-36fef3d76082"/> /* Spare node for
ITIC should
                                                                preferred node
fail */
         </rule>
         <rule id="prefered_location_iticd_2" score="100">
           <expression attribute="#uname" operation="eq"
value="hacluster3.svl.ibm.com"
            id="f8153d3d-f821-46e3-b41d-7e869e2960ec"/> /* Preferred node for
ITIC
                                                                to run */
         </rule>
```

```
        <rule id="prefered_location_iticd_3" score="-INFINITY">
          <expression attribute="#uname" operation="eq"
value="hacluster1.svl.ibm.com"
           id="7b7e7ec4-1381-47fb-afdf-732c4b180ba6"/> /* ITIC cannot
co-exist with
                                                        Maximo on this
node */
        </rule>
        <rule id="prefered_location_iticd_4" score="-INFINITY">
          <expression attribute="#uname" operation="eq"
value="hacluster4.svl.ibm.com"
           id="1fd82c28-982b-462e-b147-6726d655a87f"/> /* ITIC cannot
co-exist with
                                                        eCMDB on this
node */
        </rule>
      </rsc_location>
      <rsc_location id="location_taddm" rsc="taddm_group">
        <rule id="prefered_location_taddm_1" score="20">
          <expression attribute="#uname" operation="eq"
value="hacluster2.svl.ibm.com"
           id="b029d8a7-5a40-481f-8ebc-1168d6d76efa"/> /* Spare node for
eCMDB should
                                                        preferred node
fail */
        </rule>
        <rule id="prefered_location_taddm_2" score="100">
          <expression attribute="#uname" operation="eq"
value="hacluster4.svl.ibm.com"
           id="2cdf690e-e9c7-464e-9148-21be25565161"/> /* Preferred node for
eCMDB
                                                        to run */
        </rule>
        <rule id="prefered_location_taddm_3" score="-INFINITY">
          <expression attribute="#uname" operation="eq"
value="hacluster1.svl.ibm.com"
           id="a3065c9e-e253-4890-879f-9cf143d82fed"/> /* eCMDB cannot
co-exist with
                                                        Maximo on this
node */
        </rule>
        <rule id="prefered_location_taddm_4" score="-INFINITY">
          <expression attribute="#uname" operation="eq"
value="hacluster3.svl.ibm.com"
```

```
            id="dfd2f607-a322-483d-af67-b33b6ba3556d"/> /* eCMDB cannot
co-exist with
                                                        ITIC on this node
*/
         </rule>
       </rsc_location>
     </constraints>
   </configuration>
 </cib>
</code>
```

测试场景

我们在此处使用的示例是一个包含 3 个应用程序、3 个节点的系统，其中包含一个空闲节点。对于上述算法，我们使用包含 4 台机器的集群，其中包括 Maximo、eCMDB（Enterprise TADDM 服务器的简称）、IC（Integration Composer 的简称）以及一个被动空闲机器，所有机器都具有相同的硬件智能特性。为远程站点提供类似的 4 台机器。被动空闲机器可以运行任何 Maximo、eCMDB 或 IC 应用程序。我们采用的算法遵循以下逻辑路径：

1. 出现故障的第一台机器或第一个应用程序在空闲机器上运行。如果初始节点恢复正常，这台机器或应用程序将自动故障转移到初始节点。
2. 在出现第二次故障时，空闲机器上的应用程序优先级如下：Maximo，然后是 eCMDB，最后是 IC。Maximo 是最关键的应用程序，因为客户机应用程序希望能够随时调用它。
3. 在两个站点上遵循相同的优先级：本地和远程站点。

我们只需为 3 个不同的应用程序提供一个空闲节点。这些应用程序拥有内置的优先级。尽管所有 3 个应用程序都期望拥有高可用性，但是该可用性拥有先后次序，并且所有机器和应用程序都必须遵循这个顺序。

如果我们拥有一个必须全天候可用的应用程序，而其他应用程序没有严格的 HA 需求，那么我们至少需要一个空闲节点。如果我们拥有两台具有相同的 HA 配置文件的机器，那么我们需要两个空闲节点，依此类推。如果空闲节点上不能同时存在多个应用程序，那么所有指定的条件都为真。也就是说，将 3 个应用程序安装在同一台（空闲）机器上不会造成冲突。图 1 给出了一个包含 3 个应用程序、4 个节点的集群的示例。右侧图片是左侧图片的镜像，用于灾难恢复。这个镜像仅在左侧的所有 4 台机器都宕机之后才使用。

在下表中，Maximo (= A)、eCMDB (= B) 和 ITIC (= C) 在独立的机器上运行。我们忽略了机器名称，因为它们无关紧要，每台机器代表在其上运行的应用程序。

这些机器不运行其他任何重要的应用程序，但是指定的应用程序除外。被动空闲机器（= O）安装了 Maximo、eCMDB 和 ITIC，但是仅支持一个应用程序处于执行模式。

记住，在我们的 HA 设计中，应用程序优先级为 Maximo > eCMDB > ITIC，O 是指定的协调程序（空闲节点）。

表 1. 第一次故障

出现故障的应用程序	第一次故障之后的应用程序配置
A	A => O; B; C
B	A; B => O; C
C	A; B; C => O
O	无策略

图 2. 第一次故障

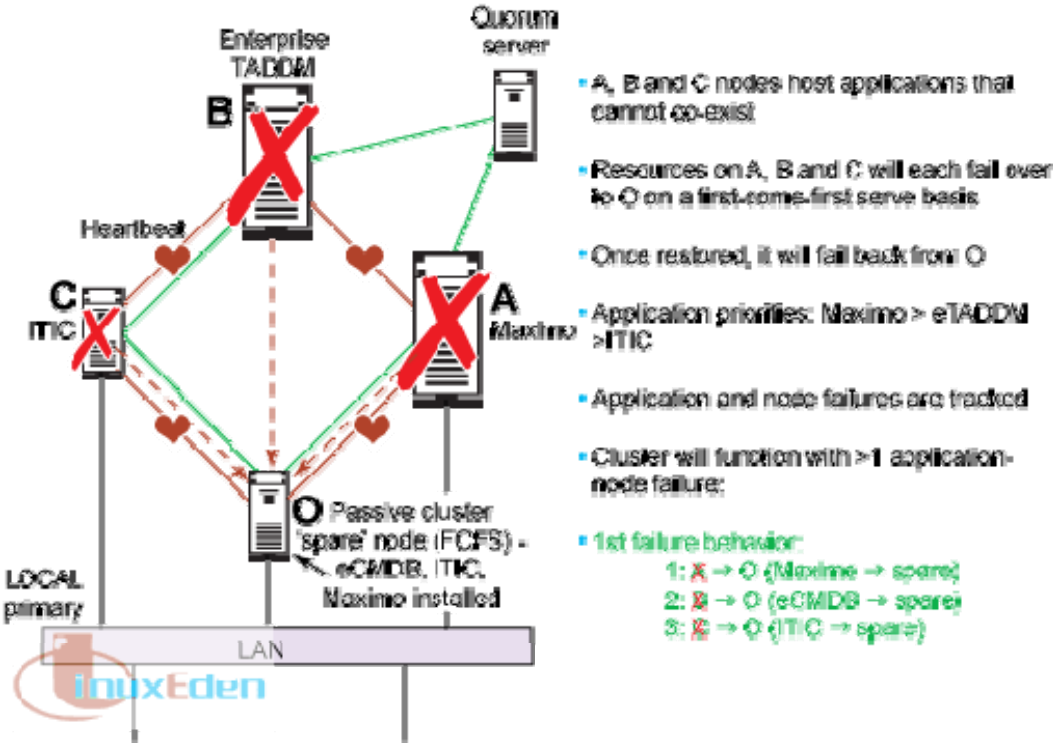


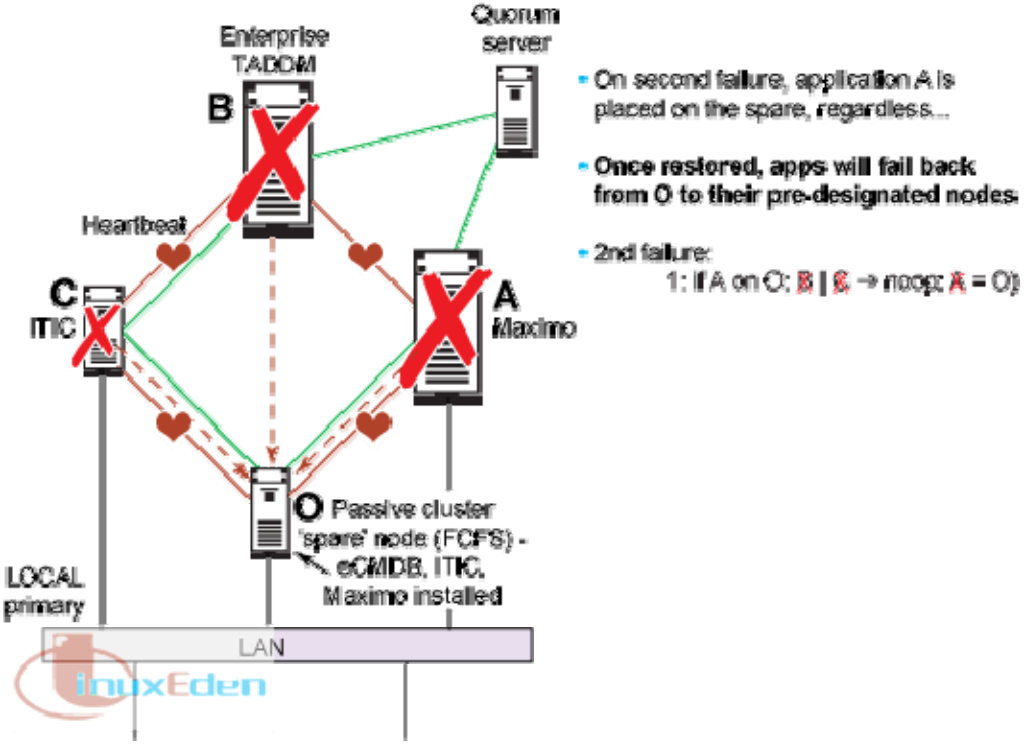
表 2. 第二次故障（A = O 表示在第一次故障之后，A 已经在 O 上运行）

出现第一次故障的机器	第二次故障之前的配置	第二次故障	第二次故障之后的配置
A	A => O; B; C	B	A = O; B unavail; C;
		C	A = O; B; C unavail
B	A; B => O; C	A	B exits; A => O; C; B unavail
		C	A; B = O; C unavail
C	A; B; C => O	A	C exits; A => O; B; C unavail

	B	C exits; A; B => O; C unavail

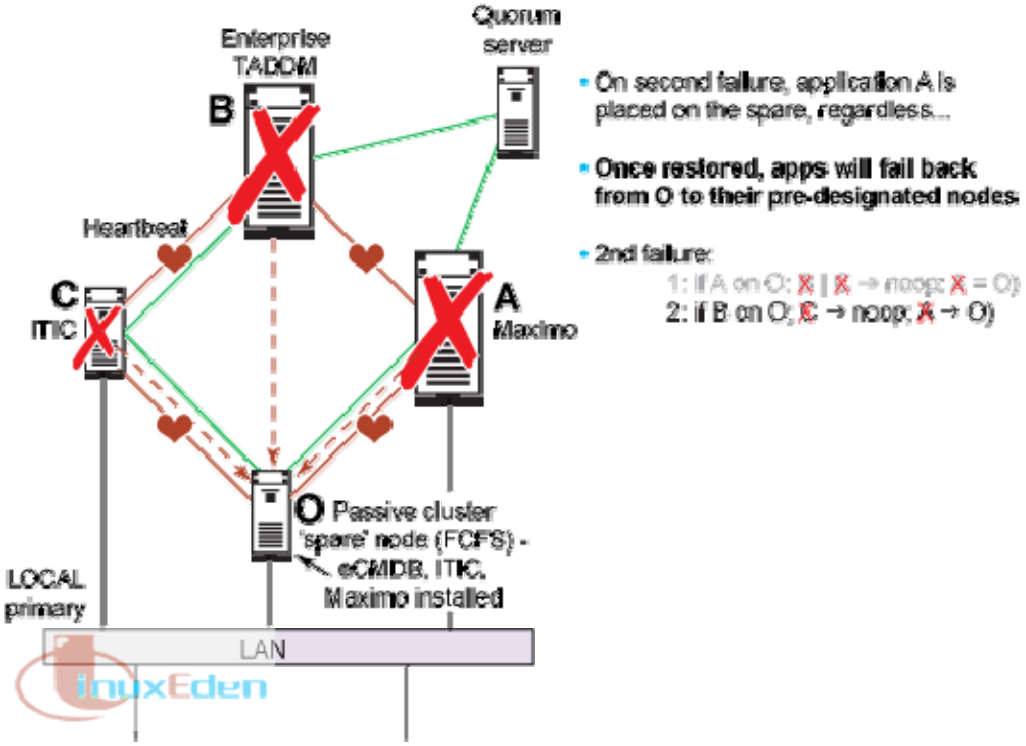
对于第二次故障，情形 1：如果 A 出现故障，它将被转移到空闲节点上。如果接下来 ITIC 或 eCMDB 出现故障，那么将不会发生任何事件。Maximo 将一直可在空闲节点上使用。

图 3. 第二次故障，情形 1



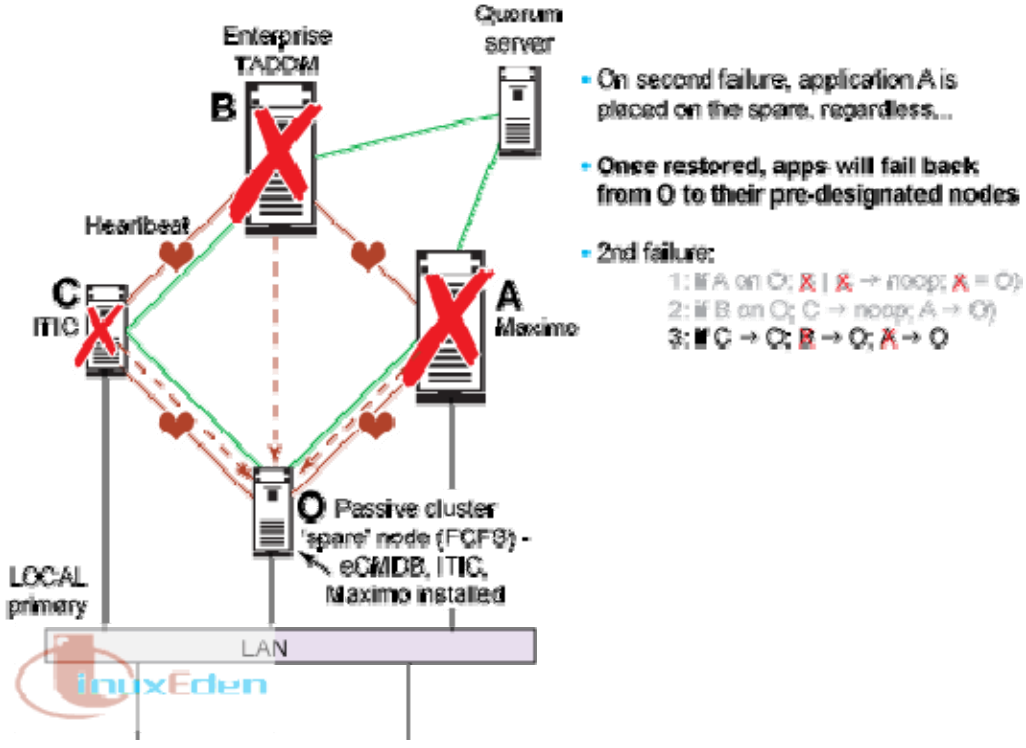
对于第二次故障，情形 2：如果 eCMDB 位于空闲节点上，那么 ITIC 随后出现故障时将不会更改现状，但是 Maximo 将从空闲节点上转储 eCMDB。

图 4. 第二次故障，情形 2



对于第二次故障，情形 3：如果 ITIC 位于空闲节点上，那么 eCMDB 在随后出现故障时将从所在位置转储 ITIC。如果接下来 Maximo 出现故障，那么它将从空闲节点转储 eCMDB。如果集群中只有空闲节点可用，那么 Maximo 将是惟一运行的应用程序。

图 5. 第二次故障，情形 3



实现小结

该实现遵循以下规则：

1. 4 节点集群中有一个被动空闲节点。
2. 在指定节点上运行 3 个应用程序。
3. 多个应用程序不能同时处于执行模式（它们是相互排斥的）。
4. 当应用程序出现故障时，它将重新启动两次，然后将故障转移到空闲节点。
5. 当节点失败时，应用程序将故障转移到空闲节点。当应用程序预先指定的节点恢复之后，应用程序将故障转移回该节点上。
6. 应用程序故障转移大约需要 45 秒。TADDM 需要更长时间。
7. 应用程序根据先到先服务的原则放置在空闲节点上。
8. 始终遵循可用性层次结构：Maximo > TADDM > ITIC。
   - 第二个应用程序/节点故障将确保 Maximo 始终可用。如果 ITIC 或 TADDM 已位于空闲节点上，它将被 Maximo 转储。
   - 执行完整的循环测试，以确定实现正确无误。
9. 针对上述行为调整手动生成和测试的配置文件。

灾难恢复

远程站点上的规则完全相同，但是没有应用程序处于运行状态。只会从原始位置复制到公共的外部磁盘。发生站点故障转移之后，执行流程为：

1. 停止所有原始机器（包括数据库）。
2. 将 VIP 与原始机器分离开来。
3. 将控制权（手动）转移到远程站点。
4. 指定远程站点中的网络磁盘作为主磁盘。
5. 使用我们的集群控制脚本启动 heartbeat，这将以与原始机器相同的顺序和优先级启动应用程序。
6. 将应用程序与网络磁盘和数据库同步。
7. 将原始机器的 VIP 分配给远程网络。
8. 响应客户机调用（与以前一样）。

结束语

本文描述了使用 Linux-HA 完成的复合应用程序的 HA 实现，它基于我们在客户需求方面的丰富经验。我们的 HA 任务涉及同一集群中具有不同优先顺序的多个应用程序。也许为每个应用程序添加一个空闲节点会更简单，但是这种解决方案成本较高。对于大多数真实的 HA 应用程序，您必须适应实际存在的经济条件和冗余，它们通常是相互排斥的。