STATEFUL SUBNET MANAGER FAILOVER IN A MIDDLEWARE MACHINE ENVIRONMENT

Inventors: Bjorn-Dag Johnsen, Oslo (NO); Line Holen, Fetsund (NO); Roy Arntsen, Oslo (NO)

Assignee: ORACLE INTERNATIONAL CORPORATION, Redwood Shores, CA (US)

Appl. No.: 13/235,113

Filed: Sep. 16, 2011

Related U.S. Application Data


ABSTRACT

A system and method can provide stateful subnet manager failover in a middleware machine environment. The system includes a policy daemon associated with each master subnet manager candidate in a subnet in the middleware machine environment. The policy daemon manages one or more policies for the subnet. The system also includes a transactional interface associated with the policy daemon co-located with a current master subnet manager. The transactional interface allows for updating the one or more policies using a policy update transaction. The policy daemon co-located with the master subnet manager operates to replicate the policy update transaction to one or more policy daemons co-located with the subnet managers that are master candidates associated with the master subnet manager, before committing the policy update transaction. Additionally, when the master subnet manager fails, the subnet managers operate to negotiate with each other and elect a new master subnet manager.
Associating a policy daemon with a master subnet manager in a subnet in a middleware machine environment, wherein the policy daemon manages one or more policies for the subnet.

Associating a transactional interface with the policy daemon, wherein the transactional interface allows for updating the one or more policies managed by the policy daemon associated with the master subnet manager using a policy update transaction.

Replicating, via the policy daemon associated with the master subnet manager, the policy update transaction to one or more subnet managers that are master candidates associated with the master subnet manager before committing the policy update transaction.
Policy Update Transaction

Network Switch 501
Policy Daemon 511
Old Master Subnet Manager A 521

Network Switch 502
Policy Daemon 512
Subnet Manager B 522

Network Switch 503
Policy Daemon 513
New Master Subnet Manager C 523

Network Switch 504
Policy Daemon 514
Subnet Manager D 524

FIGURE 5
601 Supports a core logic implementation in a subnet manager

602 Associates a policy daemon with the subnet manager, in order to inject critical policy information for the middleware machine environment into the core logic implementation in the subnet manager

603 Associates a transactional interface with the policy daemon to support transactional behavior for policy updates in the middleware machine environment

FIGURE 6
STATEFUL SUBNET MANAGER FAILOVER IN A MIDDLEWARE MACHINE ENVIRONMENT

CLAIM OF PRIORITY


COPYRIGHT NOTICE

[0002] A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

FIELD OF INVENTION

[0003] The present invention is generally related to computer systems and software such as middleware, and is particularly related to supporting a middleware machine environment.

BACKGROUND

[0004] InfiniBand (IB) Architecture is a communications and management infrastructure that supports both I/O and interprocessor communications for one or more computer systems. An IB Architecture system can scale from a small server with a few processors and a few I/O devices to a massively parallel installation with hundreds of processors and thousands of I/O devices.

[0005] The IB Architecture defines a switched communications fabric allowing many devices to concurrently communicate with high bandwidth and low latency in a protected, remotely managed environment. An end node can communicate with over multiple IB Architecture ports and can utilize multiple paths through the IB Architecture fabric. A multiplicity of IB Architecture ports and paths through the network are provided for both fault tolerance and increased data transfer bandwidth.

[0006] These are the generally areas that embodiments of the invention are intended to address.

SUMMARY

[0007] Described herein is a system and method that can provide stateful subnet manager failover in a middleware machine environment. In accordance with an embodiment, the system includes a policy daemon associated with each master subnet manager candidate in a subnet in the middleware machine environment. The policy daemon manages one or more policies for the subnet. The system also includes a transactional interface associated with the policy daemon that is co-located with a current master subnet manager. The transactional interface allows for updating the one or more policies using a policy update transaction. The policy daemon, co-located with the master subnet manager, operates to replicate the policy update transaction to one or more policy daemons co-located with the subnet managers that are master candidates associated with the master subnet manager, before committing the policy update transaction. Additionally, when the master subnet manager fails, the one or more subnet manager operate to negotiate with each other and elect a new master subnet manager.

BRIEF DESCRIPTION OF THE FIGURES

[0008] FIG. 1 shows an illustration of an exemplary configuration for a middleware machine, in accordance with an embodiment of the invention.

[0009] FIG. 2 shows an illustration of a middleware machine environment, in accordance with an embodiment of the invention.

[0010] FIG. 3 shows an illustration of a middleware machine environment that supports a policy transaction, in accordance with an embodiment of the invention.

[0011] FIG. 4 illustrates an exemplary flow chart for supporting a policy transaction in a middleware machine environment, in accordance with an embodiment of the invention.

[0012] FIG. 5 shows an illustration of a stateful subnet manager failover scenario in a middleware machine environment, in accordance with an embodiment of the invention.

[0013] FIG. 6 illustrates an exemplary flow chart for implementing a system that supports stateful subnet manager failover in a middleware machine environment, in accordance with an embodiment of the invention.

DETAILED DESCRIPTION

[0014] Described herein is a system and method for providing a middleware machine or similar platform. In accordance with an embodiment of the invention, the system comprises a combination of high performance hardware (e.g. 64-bit processor technology, high performance large memory, and redundant InfiniBand and Ethernet networking) together with an application server or middleware environment, such as WebLogic Suite, to provide a complete Java EE application server complex which includes a massively parallel in-memory grid, that can be provisioned quickly, and that can scale on demand. In accordance with an embodiment of the invention, the system can be deployed as a full, half, or quarter rack, or other configuration, that provides an application server grid, storage area network, and InfiniBand (IB) network. The middleware machine software can provide application server, middleware and other functionality such as, for example, WebLogic Server, JRockit or Hotspot JVM, Oracle Linux or Solaris, and Oracle VM. In accordance with an embodiment of the invention, the system can include a plurality of compute nodes, one or more IB switch gateways, and storage nodes or units, communicating with one another
via an IB network. When implemented as a rack configuration, unused portions of the rack can be left empty or occupied by fillers.

[0015] In accordance with an embodiment of the invention, referred to herein as "Sun Oracle Exalogic" or "Exalogic", the system is an easy-to-deploy solution for hosting middleware or application server software, such as the Oracle Middleware SW suite, or Weblogic. As described herein, in accordance with an embodiment the system is a "grid in a box" that comprises one or more servers, storage units, an IB fabric for storage networking, and all the other components required to host a middleware application. Significant performance can be delivered for all types of middleware applications by leveraging a massively parallel grid architecture using, e.g., Real Application Clusters and Exalogic Open storage. The system delivers improved performance with linear I/O scalability, is simple to use and manage, and delivers mission-critical availability and reliability.

[0016] FIG. 1 shows an illustration of an exemplary configuration for a middleware machine, in accordance with an embodiment of the invention. As shown in FIG. 1, the middleware machine 100 uses a single rack configuration that includes two gateway network switches, or leaf network switches, 102 and 103 that connect to twenty-eight server nodes. Additionally, there can be different configurations for the middleware machine. For example, there can be a half rack configuration that contains a portion of the server nodes, and there can also be a multi-rack configuration that contains a large number of servers.

[0017] As shown in FIG. 1, the server nodes can connect to the ports provided by the gateway network switches. As shown in FIG. 1, each server machine can have connections to the two gateway network switches 102 and 103 separately. For example, the gateway network switch 102 connects to the port 1 of the servers 1-14 106 and the port 2 of the servers 15-28 107, and the gateway network switch 103 connects to the port 2 of the servers 1-14 108 and the port 1 of the servers 15-28 109.

[0018] In accordance with an embodiment of the invention, each gateway network switch can have multiple internal ports that are used to connect with different servers, and the gateway network switch can also have external ports that are used to connect with an external network, such as an existing data center service network.

[0019] In accordance with an embodiment of the invention, the middleware machine can include a separate storage system 110 that connects to the servers through the gateway network switches. Additionally, the middleware machine can include a spine network switch 101 that connects to the two gateway network switches 102 and 103. As shown in FIG. 1, there can be optionally two links from the storage system to the spine network switch.

**IB Fabric/Subnet**

[0020] In accordance with an embodiment of the invention, an IB Fabric/Subnet in a middleware machine environment can contain a large number of physical hosts or servers, switch instances and gateway instances that are interconnected in a fat-tree topology.

[0021] FIG. 2 shows an illustration of a middleware machine environment, in accordance with an embodiment of the invention. As shown in FIG. 2, the middleware machine environment 200 includes an IB subnet or fabric 220 that connects with a plurality of end nodes. The IB subnet includes a plurality of subnet managers 211-214, each of which resides on one of a plurality of network switches 201-204. The subnet managers can communicate with each other using an in-band communication protocol 210, such as the Management Datagram (MAD)/Subnet Management Packet (SMP) based protocols or other protocol such as the Internet Protocol over IB (IPoIB).

[0022] In accordance with an embodiment of the invention, a single IP subnet can be constructed on the IB fabric allowing the switches to communicate securely among each other in the same IB fabric (i.e. full connectivity among all switches). The fabric based IP subnet can provide connectivity between any pair of switches when at least one route with operational links exists between the two switches. Recovery from link failures can be achieved if an alternative route exists by re-routing.

[0023] The management Ethernet interfaces of the switches can be connected to a single network providing IP level connectivity between all the switches. Each switch can be identified by two main IP addresses: one for the external management Ethernet and one for the fabric based IP subnet. Each switch can monitor connectivity to all other switches using both IP addresses, and can use either operational address for communication. Additionally, each switch can have a point-to-point IP link to each directly connected switch on the fabric. Hence, there can be at least one additional IP address.

[0024] IP routing setups allow a network switch to route traffic to another switch via an intermediate switch using a combination of the fabric IP subnet, the external management Ethernet network, and one or more fabric level point-to-point IP links between pairs of switches. IP routing allows external management access to a network switch to be routed via an external Ethernet port on the network switch, as well as through a dedicated routing service on the fabric.

[0025] The IB fabric includes multiple network switches with management Ethernet access to a management network. There is in-band physical connectivity between the switches in the fabric. In one example, there is at least one in-band route of one or more hops between each pair of switches, when the IB fabric is not degraded. Management nodes for the IB fabric include network switches and management hosts that are connected to the IB fabric.

[0026] A subnet manager can be accessed via any of its private IP addresses. The subnet manager can also be accessible via a floating IP address that is configured for the master subnet manager when the subnet manager takes on the role as a master subnet manager, and the subnet manager is unconfigured when it is explicitly released from the role. A master IP address can be defined for both the external management network as well as for the fabric based management IP network. No special master IP address needs to be defined for point-to-point IP links.

[0027] In accordance with an embodiment of the invention, each physical host can be virtualized using virtual machine based guests. There can be multiple guests existing concurrently per physical host, for example one guest per CPU core. Additionally, each physical host can have at least one dual-ported Host Channel Adapter (HCA), which can be virtualized and shared among guests, so that the fabric view of a virtualized HCA is a single dual-ported HCA just like a non-virtualized/shared HCA.

[0028] The IB fabric can be divided into a dynamic set of resource domains implemented by IB partitions. Each physical host and each gateway instance in an IB fabric can be a
member of multiple partitions. Also, multiple guests on the same or different physical hosts can be members of the same or different partitions. The number of the IB partitions for an IB fabric may be limited by the P_Key table size.

In accordance with an embodiment of the invention, a guest may open a set of virtual network interface cards (vNICs) on two or more gateway instances that are accessed directly from a vNIC driver in the guest. The guest can migrate between physical hosts while either retaining or having updated vNIC associates.

In accordance with an embodiment of the invention, switches can start up in any order and can dynamically select a master subnet manager according to different negotiation protocols, for example an IB specified negotiation protocol. If no partitioning policy is specified, a default partitioning enabled policy can be used. Additionally, the management node partition and the fabric based management IP subnet can be established independently of any additional policy information and independently of whether the complete fabric policy is known by the master subnet manager. In order to allow fabric level configuration policy information to be synchronized using the fabric based IP subnet, the subnet manager can start up initially using the default partition policy. When fabric level synchronization has been achieved, the partition configuration, which is current for the fabric, can be installed by the master subnet manager.

Policy Transaction in a Middleware Machine Environment

In accordance with an embodiment of the invention, a system and method can support a policy transaction in a middleware machine environment. The system includes a policy daemon associated with a master subnet manager in an IB subnet in the middleware machine environment. The policy daemon manages one or more policies for the IB subnet. The system also includes a transactional interface associated with the policy daemon. The transactional interface allows for updating the one or more policies using a policy update transaction. Additionally, the master subnet manager is associated with one or more subnet managers that are master candidates in the middleware machine environment. The policy daemon associated with the master subnet manager operates to replicate the policy update transaction to the one or more subnet managers before committing the policy update transaction.

FIG. 3 shows an illustration of a middleware machine environment that supports a policy transaction, in accordance with an embodiment of the invention. As shown in FIG. 3, the middleware machine environment 300 includes an IB subnet or fabric 320 that manages a plurality of end nodes. The IB subnet includes a plurality of subnet managers 321-324, each of which resides on one of a plurality of network switches 301-304. The subnet managers can communicate with each other using an in-band communication protocol such as the Internet Protocol over Infiniband (IPoIB). The subnet managers can negotiate among each other and elect a master subnet manager 321, which is responsible for configuring and managing the middleware machine environment. Additionally, the subnet managers B-D are standby master candidates in the middleware machine environment, each of which is ready to take over the master subnet manager when necessary.

In accordance with an embodiment of the invention, each network switch can connect with one or more end nodes, such as the host servers within the middleware machine environment. Both the network switch and the subnet managers residing on top of the network switch can be considered as management nodes from the perspective of a network high availability management model. The network switch can be either a leaf switch that communicates directly with the end nodes, or a spine switch that communicates with the end nodes through the leaf switches. The network switches can communicate with the host servers via the switch ports of the network switches and the host ports of the host servers. In an IB network, partitions can be defined to specify which end ports are able to communicate with other end ports.

In accordance with an embodiment of the invention, the middleware machine environment employs a fat-tree topology, which allows a small number of switches sitting at the top layers of the fat tree while maintaining a large number of end nodes as leafs of the tree.

In accordance with an embodiment of the invention, the system can provide a plurality of policy daemons 311-314, each of which is associated with a subnet manager. The policy daemon that collocates with the master subnet manager is responsible for configuring and managing the end nodes in the middleware machine environment using one or more policies. One exemplary policy managed by a policy daemon in a middleware machine environment can be a partition configuration policy. The partition configuration policy can be supplied to the subnet through an initialization policy transaction.

For example, a middleware machine environment that includes end nodes, A, B, and C can be partitioned into two groups: a Group I that includes nodes A and B and a Group II that includes node C. A partition configuration policy can define a partition update that requires deleting node B from the Group I, before adding node B into the Group II. This partition configuration policy can require that the master subnet manager not allow a new partition to add node B into Group II without first deleting nodes B from Group I. This partition configuration policy can be enforced by the master subnet manager using a policy daemon.

In accordance with an embodiment of the invention, the system can provide a transactional interface that is associated with the policy daemon. The transactional interface allows for updating the one or more policies managed by the policy daemon using a policy update transaction. The policy daemon associated with the master subnet manager can replicate the policy update transaction to the subnet manager master candidates before committing the policy update transaction. Additionally, the system provides a command interface that is responsible for providing policies to the master subnet manager.

By replicating the policy updates from the master subnet manager to the subnet manager master candidates, the system can ensure that the policies are synchronized within the middleware machine environment. When the standby subnet manager takes over and becomes the new master subnet manager, the functioning of the middleware machine environment can be uninterrupted and the communication in the middleware machine environment can maintain undisturbed. Additionally, the system can remove all stale policy information before applying the new policy or the policy updates, in order to prevent inconsistency between the master subnet manager and different instances of the subnet manager master candidates.

In accordance with an embodiment of the invention, a policy update transaction can include either a new policy or
a set of policy updates. Each policy update transaction can be represented using a unique version number. A master subnet manager can consider a policy associated with the highest version number, in its knowledge, as the current policy to be used in the middleware machine environment. In one embodiment, the system is configured so that there is only one policy update transaction in progress at any point of time in the subnet.

In accordance with an embodiment of the invention, the replication of the policy updates from the master subnet manager to the standby subnet managers fails, or alternatively the master subnet manager fails, then the policy update transaction is not committed, and the system does not apply the policy updates on the middleware machine environment in order to preserve consistency in policy configuration. Furthermore, the system allows a user or an administrator to intervene and manually set up the environment.

In accordance with an embodiment of the invention, each policy daemon can have a close relationship with the subnet manager instance on the corresponding node. The subnet manager can perform a synchronization operation with the local policy daemon whenever it becomes a master subnet manager. This ensures that the policy daemon can prepare all required current policy information that the subnet manager needs in order to initialize and maintain the state of the subnet. For example, such information includes the current partitioning configuration which can be provided as a local file.

Additionally, the various policy daemon instances can cooperate to replicate the policy information that is supposed to be shared among the subnet managers. For example, the local synchronization with the local subnet manager instance can ensure that the replication of a new version of a configuration file is complete and accurate before the master subnet manager starts to apply the new policy.

In accordance with an embodiment of the invention, when a standby subnet manager reboots, it can synchronize with the current master or other currently available master candidates for the current fabric policy.

FIG. 4 illustrates an exemplary flow chart for supporting a policy transaction in a middleware machine environment in accordance with an embodiment of the invention. As shown in FIG. 4, at step 401, a policy daemon can be associated with a master subnet manager in a subnet in the middleware machine environment, wherein the policy daemon manages one or more policies for the subnet. Furthermore, a transactional interface can be associated with the policy daemon at step 402. The transactional interface allows for updating the one or more policies managed by the policy daemon associated with the master subnet manager using a policy update transaction. Then, at step 403, the policy daemon co-located with the master subnet manager can replicate the policy update transaction to one or more policy daemons co-located with the one or more subnet managers that are master candidates before committing the policy update transaction.

Stateful Subnet Manager Failover

FIG. 5 shows an illustration of stateful subnet manager failover scenario in a middleware machine environment, in accordance with an embodiment of the invention. As shown in FIG. 5, the middleware machine environment 500 includes a plurality of network switches 501-504 together with a plurality of subnet managers 521-524 that manage a plurality of end nodes. The plurality of subnet managers can communicate with each other using an in-band communication protocol 510, such as the Internet Protocol over InfiniBand (IPoIB).

In the example as shown in FIG. 5, when an old master subnet manager A 521 fails, the rest of the subnet managers B-D can negotiate with each other and elect a new master subnet manager C, which is responsible for configuring and managing the subnet in the middleware machine environment. The new master subnet manager C can determine the most recent versions of the fabric configuration policy information along with all available subnet managers B-D. Additionally, a transaction interface 508 associated with the new master subnet manager C is used by the system to support a new policy update transaction 509.

In order to determine the current fabric configuration policy information, the system can use a quorum-based policy in the policy daemon, which specifies a minimum number of the subnet managers needed in the middleware machine environment in order to support a policy update. For example, a quorum-based policy can require more than half of all the standby subnet managers must be in synchronization. If less than half of all the standby subnet managers have the same policy, then a quorum cannot be reached, and no fabric policy changes can be implemented until either a quorum has been reached, or until a system administrator redefines the master candidate set. For example, a "split-brain" condition can be detected in a middleware machine environment with only two subnet managers. When a single point of failure disables one subnet manager, then there is only one subnet manager existing in the system and the quorum-based policy can prevent the subnet manager which detects the "split-brain" condition from taking on any master role.

In accordance with an embodiment of the invention, a policy update transaction 509 is committed only when a quorum of different policy daemons all agree on the policy update. Additionally, the policy daemon 513 can ensure that the current policy is in sync with a quorum of policy daemons before allowing a newly elected master subnet manager 523 to complete initialization of the subnet.

In accordance with an embodiment of the invention, a quorum based scheme has the advantage of being able to both implement and change a policy following one or more failures as long as a sufficient level of redundancy is provided, i.e. the system is configured with sufficient number of independent master subnet manager candidate instances.

For example, a decision about implementing, or changing and implementing, a policy can be based on an assumption that there are no conflicting policy decisions made among other potential master candidates. If only exactly half of the configured standby subnet managers are available (e.g. 2 out of a total of 4 subnet managers or 1 out of a total of 2 subnet managers), then no decision to implement or change the policy is permitted. Thus, a population of 3 or 4 master candidates can survive a single point of failure and still be able to establish a quorum that can make decisions about implementing or changing the policy. Furthermore, a population of 5 or 6 master candidates can tolerate 2 failures, and 8 master candidates can tolerate 3 failures, and so on.

In accordance with an embodiment of the invention, a consensus based scheme can be used in the system, when it is impossible to establish a quorum (or majority) following a single point of failure, for example for a configuration with only two master subnet manager candidates. The consensus
based rules can implement the current policy when at least one single master subnet manager can be established. However, in order to preserve consistency in the system, the current policy may be changed when any master subnet manager candidate is not a part of the upgrade transaction.

[0052] The advantage of the consensus based scheme is that any subnet manager that becomes the master can immediately configure the subnet based on the current policy as long as the local policy daemon can determine that the policy state reflects a committed update transaction. The drawback is that a single point of failure that makes a single subnet manager master candidate unavailable can prevent any further policy update transactions.

Implementation without Third Party Constraints

[0053] In accordance with an embodiment of the invention, an implementation of the system as described above requires only a minimal change in an existing subnet manager implementation, and allows the subnet manager implementation to be based on third party source code, such as open source shared code. The system allows the handling of stateful failover to be implemented without the open source constraints and also can be independent of the IB fabric that the subnet manager relates to.

[0054] FIG. 6 illustrates an exemplary flow chart for implementing a system that supports stateful subnet manager failover in a middleware machine environment, in accordance with an embodiment of the invention. As shown in FIG. 6, at step 601, the subnet manager employs a core logic implementation. The core logic of the subnet manager can implement an IB standard such as the OpenSM standard, which only supports stateless failover. Furthermore, at step 202, a policy daemon can be associated with the subnet manager in order to inject critical policy information for the middleware machine environment into the core logic implementation in the subnet manager. The core logic implementation in the subnet manager may not be aware of the policy daemon. The implementation of the policy daemon requires only minimal change to the core logic implementation, and can be independent of and separated from the core logic implementation in the subnet manager. Additionally, a transactional interface can be associated with the policy daemon, at step 203. The transactional interface allows transactional behavior for policy updates in the middleware machine environment. The transactional behavior can ensure full ACID (atomicity, consistency, isolation, durability) properties for the policy updates without a need of including the transaction logic within the core logic implementation in the subnet manager.

[0055] The present invention may be conveniently implemented using one or more conventional general purpose or specialized digital computer, computing device, machine, or microprocessor, including one or more processors, memory and/or computer readable storage media programmed according to the teachings of the present disclosure. Appropriate software coding can readily be prepared by skilled programmers based on the teachings of the present disclosure, as will be apparent to those skilled in the software art.

[0056] In some embodiments, the present invention includes a computer program product which is a storage medium or computer readable medium (media) having instructions stored thereon/in which can be used to program a computer to perform any of the processes of the present invention. The storage medium can include, but is limited to, any type of disk including floppy disks, optical discs, DVD, CD-ROMs, microdrive, and magneto-optical disks, ROMs, RAMs, EPROMs, EEPROMs, DRAMs, VRAMs, flash memory devices, magnetic or optical cards, nanosystems (including molecular memory [Cs]), or any type of media or device suitable for storing instructions and/or data.

[0057] The foregoing description of the present invention has been provided for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Many modifications and variations will be apparent to the practitioner skilled in the art. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, thereby enabling others skilled in the art to understand the invention for various embodiments and with various modifications that are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the following claims and their equivalence.

What is claimed is:

1. A system for supporting policy transaction in a middleware machine environment, comprising:
   one or more microprocessors;
   a policy daemon, running on the one or more microprocessors, associated with a master subnet manager in a subnet in the middleware machine environment, wherein the policy daemon manages one or more policies for the subnet;
   a transactional interface associated with the policy daemon, wherein the transactional interface allows for updating the one or more policies managed by the policy daemon associated with the master subnet manager using a policy update transaction, and
   wherein the master subnet manager is associated with one or more subnet managers that are master candidates in the subnet, and the policy daemon associated with the master subnet manager operates to replicate the policy update transaction to the one or more subnet managers before committing the policy update transaction.

2. The system according to claim 1, wherein:
   the subnet is Infiniband (IB) subnet that includes a plurality of management nodes connecting with a plurality of host servers.

3. The system according to claim 2, wherein:
   the plurality of management nodes include one or more network switches, wherein each said subnet manager resides on a network switch.

4. The system according to claim 1, wherein:
   each said subnet manager is associate with a different policy daemon.

5. The system according to claim 4, wherein:
   the policy update transaction is committed only when a quorum of said different policy daemons agrees.

6. The system according to claim 1, wherein:
   when the master subnet manager fails, the one or more subnet managers operate to negotiate with each other and elect a new master subnet manager, which is responsible for configuring and managing the middleware machine environment.

7. The system according to claim 1, wherein:
   the subnet uses an in-band communication protocol to connect the master subnet manager with the one or more subnet managers.
8. The system according to claim 1, wherein:
   a said policy is a partition policy that can define a partition
   configuration in the subnet, and wherein the partition
   policy can be supplied to the subnet through an initial-
   ization policy transaction.
9. The system according to claim 1, further comprising:
   a command interface that is responsible for providing poli-
   cies to the master subnet manager via the transactional
   interface.
10. The system according to claim 1, wherein:
    the master subnet manager can use a default partitioning
    policy for initialization when no partitioning policy is
    specified.
11. The system according to claim 1, wherein:
    the master subnet manager ensures that functioning of the
    middleware machine environment is not be interrupted
    when a standby subnet manager takes over and becomes
    a new master subnet manager.
12. The system according to claim 1, wherein:
    all stale policy information can be removed before apply-
    ing the new policy or the policy updates.
13. The system according to claim 1, wherein:
    the policy update transaction can include either a new
    policy or a set of policy updates, and the policy update
    transaction can be represented using a unique version
    number.
14. The system according to claim 14, wherein:
    the master subnet manager considers a policy associated
    with a highest version number as the current policy used
    in the middleware machine environment, and the subnet
    allows one policy update transaction in progress at any
    point of time in the subnet.
15. The system according to claim 1, wherein:
    the policy daemon ensures that current policy is in synch
    within a quorum of policy daemons before allowing a
    newly elected master subnet manager to complete ini-
    tialization of the subnet.
16. The system according to claim 1, wherein:
    a consensus based scheme is used when it is impossible to
    establish a quorum following a single point of failure,
    wherein the consensus based rules can implement a cur-
    rent policy when at least one single master subnet man-
    ager is established and the current policy can not be
    changed when any master subnet manager candidates is
    not a part of the upgrade transaction.
17. The system according to claim 1, wherein:
    the subnet manager is implemented with a core logic based
    on third party source code.
18. The system according to claim 17, wherein:
    the policy daemon can inject critical policy information for
    the middleware machine environment into the core logic
    implementation in the subnet manager.
19. A method for supporting policy transaction in a middle-
    ware machine environment, comprising:
    associating a policy daemon running on one or more micro-
    processors with a master subnet manager in a subnet in
    the middleware machine environment, wherein the
    policy daemon manages one or more policies for the
    subnet;
    associating a transactional interface with the policy dae-
    mon, wherein the transactional interface allows for
    updating the one or more policies managed by the policy
    daemon associated with the master subnet manager
    using a policy update transaction; and
    replicating, via the policy daemon associated with the mas-
    ter subnet manager, the policy update transaction to one
    or more subnet managers that are master candidates
    associated with the master subnet manager before com-
    mitting the policy update transaction.
20. A machine readable medium having instructions stored
    thereon that when executed cause a system to perform the
    steps of:
    associating a policy daemon running on one or more micro-
    processors with a master subnet manager in a subnet in
    the middleware machine environment, wherein the
    policy daemon manages one or more policies for the
    subnet;
    associating a transactional interface with the policy dae-
    mon, wherein the transactional interface allows for
    updating the one or more policies managed by the policy
    daemon associated with the master subnet manager
    using a policy update transaction; and
    replicating, via the policy daemon associated with the mas-
    ter subnet manager, the policy update transaction to one
    or more subnet managers that are master candidates
    associated with the master subnet manager before com-
    mitting the policy update transaction.
* * * * *