Domains

Seminar on High Availability and Timeliness in Linux

Zhao, Xiaodong
March 2003
Department of Computer Science
University of Helsinki

1. Introduction

The Hardware Platform Interface (HPI) is developed by the Service Availability Forum. It is designed to monitor and control of the actual platform hardware. It represents the platform-specific characteristics of the system in an abstract model. This kind of model provides high-availability system platforms with built-in redundancy and extensive management capabilities. [SAF02a] In this model there are two kinds of representations, "physical view" and "management view". The physical view identifies each component in the system with an entity identifier, which describes the type of hardware component and the entity instance that allows multiple occurrences of a particular hardware component to be distinguished. In the management view, the management capabilities are modeled in the "management instruments" interface, which are addressed by a hierarchical address consisting of a "Domain ID", "Resource ID", and "Instrument ID". [SAF02a]

A resource is a collection of management instruments associated with one or more entities in the system. If the management instruments share common accessibility in the system, they belong to the same resource. A domain is a collection of resources. The resources can be shared by more than one domain. Users who can access the domain are also able to access all the management instruments contained in a resource. The physical view of the systems is overlaid on the management view by associating entity paths with each management instrument. [TAD02] Management instruments are defined as sensors, controls, entity inventory repositories and watchdog timers.

The overall management view is as figure 1. The responsibilities of domains are discovery of resources, session management, event generation and log management, which are described as following.

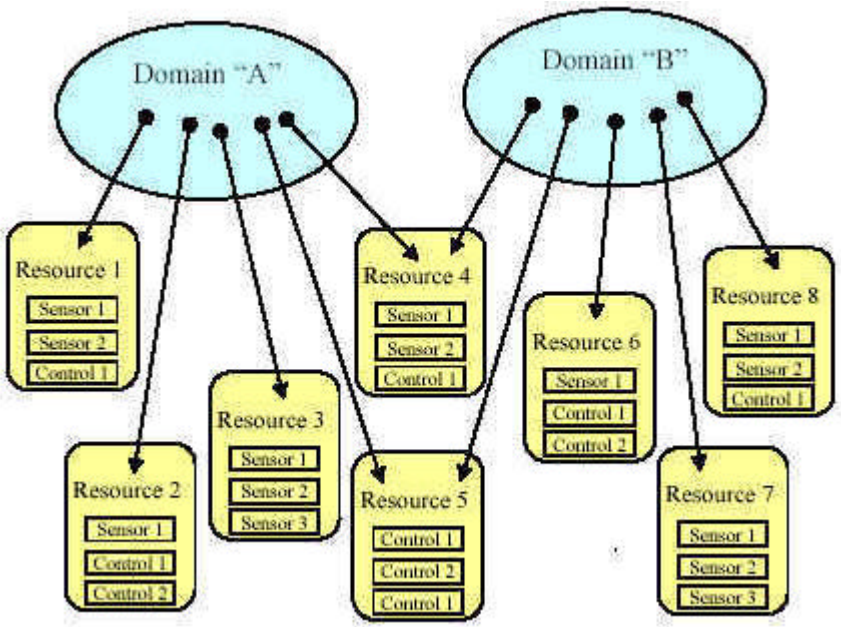

Fig. 1: The Management View

2. Discovery of Resources

The interface can describe what platform capabilities it offers for the user. By this feature, the user also can discover the resources and entities under management control. The discovery of resources is accomplished by accessing two data structures: the Resource Presence Table (RPT) associated with a domain, and the Resource Data Records (RDRs) associated with a resource. [SAF02b]

The RPT is maintained for each domain defined in the interface. In RPT there is an entry for each resource that is a member of that domain. A user may see the part of the overall system by reading the RPT. The user can only read the part that is permitted by the domain. If the domain does not allow the permission on a part, the user cannot get visibility to that part. An RPT also contains entries that reference additional domains available through the interface. The user may discover all the resources available in the platform by these entries even if those resources are not included in a single domain. There is one specification defined to support the full system discovery.

This specification requires that the HPI implementation include domain reference entries in RPTs. If a user accesses a special "default domain" defined in the specification, it accesses all the domains referenced in the RPT of that domain and continues this process in all those domains. [SAF02b] Therefore all domains available in the system will be discovered. Each entry describing a resource will include the resource ID for the resource. Each entry describing another domain will include the domain ID for the domain. If a resource is also a domain, the PRT entry will contain both its resource ID and its domain ID.

A user can learn the resources accessible through a domain. After discovering each resource, the user can then access RDRs maintained by the resource. The records describe each of the management instruments available in the resource. The RDRs are different for sensors, controls, entity inventory repositories and watchdog timers. [SAF02b] While reading RDRs, the user software will encounter entity paths for each management instrument. The service availability middleware or other carrier-grade software will need to create a physical view of the entire system. By collecting the entity paths, discarding duplicates, the physical view of the system can be constructed.

3. Session Management

All accesses to the HPI by users are processed in relation to a context called a "session". Sessions are managed on a domain-by-by basis. The session management functions deal with opening and closing sessions for the HPI implementation. They provide the scope for accessing resources and retrieving events. [SAF02b]

When a session is initiated, a domain identifier is provided. The subsequent function calls on that session will access resources in that domain exclusively. The access on the session to the resources can be controlled and restricted. Once a session is opened,

it is tightly associated with a single domain. In addition to identifying a domain, a session also may contain an event queue, and implementation-specific permissions set when session is opened. Therefore, most function calls require the use of a session handle to identify the session context for the caller.

If a session to a domain is established, the user software can use function calls in the interface to discover all the resources that are members of the domain by reading the domain RPT. By reading the RDRs in each discovered resource, it can also discover all the management instruments and entities visible and accessible via that domain. [SAF02b] If the domain RPT contains references to other domains, the user software may open additional sessions to those domains to further discover other parts of the system. It may encounter the same resource in more than one domain during this process because one resource may be shared by more than one domain.

4. Event Generation and Log Management

Users need subscribe to receive events from the HPI on a domain-by-domain basis. The HPI sets up an event queue for the user after subscription. It also places a copy of all events related to any resource that is a member of that domain on the queue as they are generated. A user may read events from the queue via a blocking or non-blocking call. [SAF02b] If a user process makes a blocking call to receive an event, that process will wait until an event is received. It will be woken up to process the received event.

The user should be aware of the initial state of the system to operate by receiving and responding to events. The HPI provides a special feature that allows the user to learn this initial state via processing event message. When the user subscribes to receive events from a domain, a flag may be set that requests the HPI place events on the user's event queue for all active alarms in the system. It effectively recreates events

that occurred prior to the subscription request. The user needs to know about if it is not going to poll all sensors for their current state. [SAF02b]

In addition to event generation, a domain controller maintains a system event log for all events generated by resources in that domain. Individual resources may maintain system event logs. If a local system event log is supported, the event log capability set in its RPT table entry will be indicated.

Each event log can have its own time clock, which is used for setting timestamp on log entries to that log. [SAF02b] Therefore, event log can be implemented on separate hardware units that may have their own time clocks. The user should be aware that log entries from a given log are time-stamped using that log's time clock, which may be different to other time clocks.

The event log is implementation-specific. All system event logs, either the domain system event log, or a resource system event log may be managed by the user through the HPI functions. [SAF02b] The activities involved in the log management include reading, writing, clearing records and setting the timestamp clock, etc.

5. Hot Swap Capabilities

Managing hot swappable devices and a changing platform configuration is an important feature for the Service Availability Forum Platform Interface. A component that may be added or removed from the system is called a "Field Replaceable Unit" or "FRU" in the specification. [SAF02a] FRU allows the interface to dynamically adjust as the hardware platform configuration changes.

When a FRU is inserted in the system, the HPI implementation will add a corresponding resource to at least one domain. It will generate an appropriate event

message. User software can then access the RDRs in that newly added resource to discover the capabilities of the FRU. Service Availability middleware can dynamically incorporate it into its system model if a new type of FRU is added to the system. When a FRU is removed from the system, the corresponding resource is removed from any domains in which it is a member. The appropriate events are sent. User software is aware that all the management instruments hosted by that resource are no longer available in the system, and the entities that they are associated with are removed.

A simplified hot swap model is illustrated in figure 2.

Fig. 2: Simplified Hot Swap Model

6. Fault Tolerance

6.1 Redundancy

Redundancy means that when one component fails in the system, another one will be available and ready to continue processing. [SAF02c] The most fundamental capability provided by the hardware of high availability fault-managed systems is fault domains. [HAF01] A single computer is made up of multiple redundant fault domains. The hardware design is to make sure the computer continues to provide full service even if any of its fault domains fail. In any high availability system, fault domains must be identified. For each fault domain, the required redundancy must also be identified. In a clustered system, fault domains are complete computer nodes.

Redundancy indicates there are enough nodes that support the minimum required performance of the system when any one of the nodes is out of service.

6.2 Failure Detection

Failure detection ability is critical in the fault domains. An example is fail-safe behavior, where a fault domain contains a self-checking capability that causes it to shutdown when a fault is detected. The resulting shutdown is then observed by other parts of the system.

When a system contains fault domains that are effectively in a standby mode, there is a need for detection of latent faults in these domains. That is, if the primary failure detection mechanism is to observe normal operating behavior, the hardware may need to provide a separate mechanism to detect faults in fault domains.

6.3 Fault Domain Diagnosis

If a failure of fault domain occurs but it is not evident which of the several redundant fault domains has failed, a diagnostic function must be provided to determine where the actual failure exists. Diagnosis of failures may occur through diagnosis capabilities of the hardware components themselves or through capabilities of management software that can analyze a variety of actions to determine which hardware component has failed. In a system with nested fault domains, diagnosis may aid in converting a detected failure of the larger fault domain to a failure of the nested fault domain. Additional diagnosis capabilities are also useful to repair a fault domain. For example, if an I/O controller fails, after the system isolates and recovers from the failure, it would be desirable to have the platform management system order the controller to run a self-test to determine if the failure was transient or permanent. If it is transient, the I/O controller can be immediately reintegrated. If it is permanent, a manual fix is necessary. [HAF01]

6.4 Fault Domain Isolation

Isolation means the action to prevent a fault from affecting other fault domains. This ability is critical in the design of a fault-managed system. It is an integral part of the design of a hardware component. For example, power supplies include circuits that monitor output voltages being produced, and quickly shut the supply off if voltages venture out of specifications. In other cases, another part of the system may need to initiate an action to isolate a fault domain. Even when isolation is automatic by design, it can be highly desirable, as a further safety feature, for any fault domain to be able to isolate itself from the rest of the system upon request from a platform management system. [HAF01]

It is possible that a fault domain misbehaves. Therefore multiple levels of fault isolation capability are needed. For example, if a host processor is not responding, it may be ordered to execute a system-reset operation. If this still does not clear the problem, it may be ordered to power itself off. Similarly, if a particular I/O controller has failed in a system, it may be ordered to isolate from the I/O bus. If this does not work, a second level of isolation may be to isolate a slot on the back plane, or even an entire I/O bus segment.

6.5 Fault Domain Failure Recovery

The System should be able to recover from the failure of a fault domain. The basic hardware capability is to be reconfigured as required for the system to continue to provide its services using the remaining functional fault domains. The specific requirements of the hardware will be dependent on details of the system redundancy strategies employed. As systems recover from failures of fault domains, a common problem is the preservation or migration of system state information in failed system components.

System state information is transferred to backup fault domains via management middleware or application software. Hardware may support this operation by providing specific capabilities to perform hot, warm, or cold restarts.

Hot restart - system reset operation clears and a mask all interrupts, set program counter and all CPU registers to pre-determined values while not altering any system RAM. It then begins operation.

Warm restart - system reset operation clears and masks all interrupts, sets program counter and all CPU registers to pre-determined values (perhaps different from hot restart) while not altering certain parts of system RAM containing in-memory databases. It then begins software reboot operation.

Cold restart - system reset operation clears and masks all interrupts, sets program counter and all CPU registers to pre-determined values (perhaps different from hot or warm restart) while clearing all system RAM. It then begins software boot operation. [HAF01]

6.6 Fault Domain Repair

The ability to repair failed fault domains is one of the most complex features of high availability systems. It depends on the design of the fault domains. Because of the complexity, it is usual that the requirement for on-line repair of fault domains is a major driver in the system architecture and the identification of fault domain in the first place.

If a system has different kinds of fault domains, it will require more specific hardware capabilities. Those capabilities can be:

- The basic ability to hot-swap fault domains
- Notification to other parts of the system that a system configuration has changed

- Guidance to service personnel to help ensure correct repair actions
- On-line firmware upgradability
- Maintenance of a system inventory.

7. Conclusion

In high-availability system, domains are represented in the management view. A domain is a collection of resources. Resources can be shared by more than one domain. The responsibilities of domains are discovery of resources, session management, event generation and log management. Hot swap capability is introduced in high-availability system. Platform is able to remove and replace failed components. When a component "Field Replaceable Unit (FRP)" is inserted into the system, the HPI implementation will add a corresponding resource to at least one domain. When a FRU is removed from the system, the corresponding resource is removed from any domains in which it is a member. The most fundamental capability provided by the hardware of high availability fault-managed systems is fault domains. The fault domains possess capabilities such as redundancy, failure detection, diagnosis, failure isolation, failure recovery, and repair.

Reference

[HAF01] High Availability Forum. Providing Open Architecture High Availability Solutions. 2001.
ftp://download.intel.com/platforms/applied/eiacomm/papers/ha_solutions.pdf

[SAF02a] Service Availability Forum. Whitepaper, 2002.
http://www.saforum.org/specification/sepcification_white_paper.pdf

[SAF02b] Service Availability Forum. Hardware Platform Interface Specification, 2002. http://www.saforum.org

[SAF02c] Service Availability Forum. Standards for a Service Availability Solution. 2002. http://www.saforum.org/about/solution_backgrounder.pdf

[TAD02] Timo Jokiaho and Dave Penkler. The Service Availability Forum Platform Interface. RTC Magazine, Volume 6, Number 12, December 2002. http://www.rtcmagazine.com/pdfs/2002/12/rtc12_techfocus.pdf