PCIe Errors High Level Design

Requirement

1.0 Read Device Status from PCI config space.
2.0 Read AER errors from PCI config space for devices that support this information.
3.0 Dispatch notification in case any error is found or cleared.
4.0 Retrieve AER errors from log file using regular expressions.
5.0 Read interval should be configurable.

Overview

The purpose of this feature is to monitor and report PCI Express errors. There are two mechanisms for error handling. First is base line which is mandatory for every PCIe device, but provides only limited information as there are only four error types. It resides in Device Status register of PCI Express capability. The second is extended capability with Advance Error Reporting. It can provide detailed information about errors set on device. Its occurrence is optional, and not every device provides this extended information.

The example of Device Status and AER registers obtained with `lspci`:

Capabilities: [a0] Express (v2) Endpoint, MSI 00

DevSta: CorrErr+ UncorrErr- FatalErr- UnsupReq+ AuxPwr- TransPend-

Capabilities: [100 v2] Advanced Error Reporting

UESta: DLP- SDES- TLP- FCP- CmpltTO- CmpltAbrt- UnxCmplt- RxOF- MalITLP- ECRC- UnsupReq- ACSViol-

UEMsk: DLP- SDES- TLP- FCP- CmpltTO- CmpltAbrt- UnxCmplt- RxOF- MalITLP- ECRC- UnsupReq- ACSViol-

UESvrt: DLP+ SDES+ TLP- FCP+ CmpltTO- CmpltAbrt- UnxCmplt- RxOF+ MalfTLP+ ECRC- UnsupReq- ACSViol-

CESta: RxErr- BadTLP- BadDLLP- Rollover- Timeout- NonFatalErr-

CEMsk: RxErr- BadTLP- BadDLLP- Rollover- Timeout- NonFatalErr+

The host system can natively handle Advanced Error Reporting, in such case kernel reports errors and they are logged into syslog and kern.log. The native error handling requires ACPI _OSC (Advanced Configuration and Power Interface _Operating System Capabilities) support from BIOS. If the _OSC method grants control to operating system, the PCIe AER native control is enabled. A lot of firmwares don't provide _OSC support while they use PCI Express. More can be read [here](#) and [here](##). The native control can also be enforced by kernel boot parameter `pcie_ports=native`. Below is the example of PCIe error from syslog.

```
kernel: [107476.334283] pcieport 0000:00:03.0: AER: Uncorrected (Fatal) error received: id=0300
kernel: [107476.334296] i40e 0000:03:00.0: PCIe Bus Error: severity=Uncorrected (Fatal), type=Unaccessible, id=0300(Unregistered Agent ID)
kernel: [107476.345636] i40e 0000:03:00.0: broadcast error_detected message
kernel: [107476.345640] i40e 0000:03:00.0: i40e_pci_error_detected: error 2
kernel: [107477.360026] pcieport 0000:00:03:0: Root Port link has been reset
kernel: [107477.360032] i40e 0000:03:00.0: broadcast slot_reset message
kernel: [107477.360164] i40e 0000:03:00.0: broadcast resume message
kernel: [107477.604466] i40e 0000:03:00.0: AER: Device recovery successful
```

Error categories

The PCIe errors fall into one of the following categories:

**Corrected errors**

They may have an impact on performance (like latency, bandwidth), but no data/information is lost and PCIe fabric remains reliable. Such errors are corrected by hardware and no software intervention is required.

**Uncorrected Non-Fatal errors**
No impact on integrity of the PCI Express fabric, but data/information is lost. Non-fatal errors are corrupted transactions that can't be corrected by PCIe hardware. However, the PCI Express fabric continues to function correctly and other transactions are unaffected, only particular transaction is affected. Recovery from a non-fatal error depends on device-specific software associated with the requester that initiated the transaction.

Uncorrected Fatal errors
Impact on integrity of the PCI Express fabric i.e. PCIe link is no more reliable and data/information is lost. Recovery from fatal errors is done by resetting the component and link.

Errors list

Base Line errors
Correctable Error
Non-Fatal Error
Fatal Error

Unsupported Request (belongs to Uncorrected Non-Fatal category)

AER Correctable
Receiver Error Status
Bad TLP Status
Bad DLLP Status
REPLAY_NUM Rollover
Replay Timer Timeout
Advisory Non-Fatal
Corrected Internal
Header Log Overflow

AER Uncorrectable
Data Link Protocol
Surprise Down
Poisoned TLP
Flow Control Protocol
Completion Timeout
Completer Abort
Unexpected Completion
Receiver Overflow
Malformed TLP
ECRC Error Status
Unsupported Request
ACS Violation
Internal
MC blocked TLP
Atomic egress blocked
TLP prefix blocked

Note: the severity of uncorrected errors (Fatal or Non-Fatal) depends on severity flag from corresponding register. The severities can differ and are configured per device.

Note: Every AER error has corresponding bit in Mask register. If bit is set the error can be ignored.
Design

PCIe errors plugin

The plugin creates the list of available PCIe devices using sysfs access to PCI devices and their config space. The list is enumerated on every interval and config space is polled to read available errors register. If new error is set the plugin sent notification. The type_instance and severity depends on error category. In case the error that was previously set is cleared the notification is sent with ‘NOTIF_OKAY’ severity. The flow for setting notification severity is on figure 11.
The plugin is able to parse AER errors from log file. By default the source is syslog and error notification is sent with type_instance corresponding to error category. It is possible to set different log file and error syntax with use of plugin configuration.

Considerations

Configuration Considerations

To collect information about errors, the collectd pcie errors plugin uses system access to pci devices config space and parse kernel events in log. The list of devices depending on configuration is obtained from “/sys/bus/pci/devices/” or “/proc/bus/pci/devices”. The default configuration is equivalent to below parameters:

```
<Plugin pcie_errors>
    Source "sysfs"
</Plugin>
```
AccessDir "/sys/bus/pci"
ReportMasked false
PersistentNotifications false
FirstFullRead false
LogFile "/var/log/syslog"

<MsgPattern "AER">
  <Match>
    Name "aer error"
    Regex "AER:.*error received"
    SubmatchIdx -1
  </Match>
  <Match>
    Name "incident time"
    Regex "(... ..;...). * pcieport.*AER"
    IsMandatory false
  </Match>
  <Match>
    Name "root port"
    Regex "pcieport (.): AER:"
  </Match>
  <Match>
    Name "device"
    Regex "([0-9a-F:\.\.*]: PCIe Bus Error"
  </Match>
  <Match>
    Name "severity"
    Regex "severity=[\^,\w]*"
  </Match>
  <Match>
    Name "error type"
    Regex "type=(.*)," 
    IsMandatory false
  </Match>
  <Match>
    Name "id"
    Regex ", id=(.*)"
  </Match>
</MsgPattern>
</Plugin>
Configuration options include:

**Source “sysfs”|“proc”|“logfile”**

Get the list of devices and read errors from /sys or /proc. In case of “logfile” parse the log file with use of message log parser utility. File location can be set with **LogFile** option.

**ReportMasked true|false**

If set to true the notification is dispatched also for errors with bit set in Mask register. The default value is false and masked errors are ignored. It takes effect only if **Source** is set to “sysfs” or “proc”.

**PersistentNotifications true|false**

Set the notifications to be sent persistently at every read callback while the error is set in pci config space. If false only the change of status to set/clear is notified. **Note**: at first read every error is considered new and set notification is dispatched. It takes effect only if **Source** is set to “sysfs” or “proc”.

**LogFile “/var/log/syslog”**

Optional parameter to set custom log file location. It takes effect only if **Source** is set to “logfile”.

**<MsgPattern “name”>**

Optional configuration which can be used to set custom regular expressions for log parser utility. It takes effect only if **Source** is set to “logfile”. The first and last patterns are mandatory and mark begin and end of the message. There are two special match names that should be present in the message pattern definition: “device” and “severity”. They are used to set plugin instance, severity and type instance for notification. More details are provided with collectd configuration manual and default config file.

**Note** that this parameter is only optional and if omitted the plugin uses the optimal default patterns for AER messages. It is introduced only to increase plugin flexibility and shouldn’t be used without specific reason.

**Deployment Considerations**

**API/GUI/CLI Considerations**

**Equivalence Considerations**

**Security Considerations**

**Alarms, events, statistics considerations**

**Redundancy Considerations**

**Performance Considerations**

**Testing Consideration**

**PCIe AER errors can be injected with use of “aer-inject” tool.**

**Other Considerations**

**Impact**

The following table outlines possible impact(s) the deployment of this deliverable may have on the system.

<table>
<thead>
<tr>
<th>Ref</th>
<th>System Impact Description</th>
<th>Recommendation / Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Key Assumptions**

The following assumptions apply to the scope specified in this document.

<table>
<thead>
<tr>
<th>Ref</th>
<th>Assumption</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. Plugin is intended to be used only on little-endian platforms. If the source is set to "sysfs" or "proc" the data is taken from PCI config space which is inherently little-endian.

Key Exclusions

None.

Key Dependencies

The following table outlines the key dependencies associated with this deliverable.

<table>
<thead>
<tr>
<th>Ref</th>
<th>Dependency</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Collectd message log parser utility.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Issues List

None.