Multiprocessing with UEFI

Daryl McDaniel
Sr. Systems Engineer
Software and Solutions Group

15 June, 2010
## Common Terms

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BSP</strong></td>
<td>BootStrap Processor</td>
</tr>
<tr>
<td><strong>AP</strong></td>
<td>Application Processor</td>
</tr>
<tr>
<td><strong>Framework</strong></td>
<td>Core firmware services supporting drivers and protocols.</td>
</tr>
<tr>
<td><strong>GUID</strong></td>
<td>128bit Globally Unique IDentifer</td>
</tr>
<tr>
<td><strong>Protocol</strong></td>
<td>A set of related services, named by a GUID and encapsulated within a structure. Similar to a DLL or C++ Class.</td>
</tr>
<tr>
<td><strong>Boot Services</strong></td>
<td>UEFI Services supporting drivers and OS boot operations.</td>
</tr>
<tr>
<td><strong>Runtime Services</strong></td>
<td>UEFI services available during OS runtime.</td>
</tr>
</tbody>
</table>
Multi-Processor Programming

• Platform Control
  – Enumerate Processors
  – Start / Stop / Reset Individual Processors

• Resource Exclusivity
  – Semaphores & Mutexes
  – Atomic Operations
    > Increment / Decrement
    > Test & Set / Compare & Exchange

• Inter-Processor Communications
Platform Control

- FrameworkMpService Protocol
  - GetGeneralMPInfo
  - GetProcessorContext
  - StartupAllAPs
  - StartupThisAP
  - SwitchBSP
  - SendIPI
  - EnableDisableAP
  - WhoAmI

- Startup functions are blocking.

- Timeout returns control to BSP, AP is unaffected.

- (PI) MpService Protocol
  - GetNumberOfProcessors
  - GetProcessorInfo
  - StartupAllAPs
  - StartupThisAP
  - SwitchBSP
  - EnableDisableAP
  - WhoAmI

- Startup functions may be either blocking or non-blocking.

- Timeout terminates the AP.
Resource Exclusivity

• Synchronization Library
  – GetSpinLockProperties( void)
  – InitializeSpinLock( &SpinLock)
  – AcquireSpinLock( &SpinLock)
  – AcquireSpinLockOrFail( &SpinLock)
  – ReleaseSpinLock( &SpinLock)
  – InterlockedIncrement( &Value)
  – InterlockedDecrement( &Value)
  – InterlockedCompareExchange32(&Value, Comp, Xchg)
  – InterlockedCompareExchange64(&Value, Comp, Xchg)
  – InterlockedCompareExchangePointer(&Value, &Comp, &Xchg)
Inter-Process Communication

- **Root Task**: Initialize To Data Consumed=0
- **Ready**: Start Client Task
- **Result**: Test Data Consumed=0, Set Data Present=1
- **Client Task**: Write Result

**Producer**
- Test Data Present=1
- Read Result
- Set Data Consumed=0

**Consumer**
- NO
StartCore Structure

- **Library Functions**
  - StrDecimalToUintn
  - CpuPause

- **UEFI Functions**
  - LocateProtocol
  - CreateEvent
  - CloseEvent

- **MpService Functions**
  - GetNumberOfProcessors
  - GetProcessorInfo
  - StartupThisAp
GetMpInfo

// Find the MP Services Protocol
Status = gBS->LocateProtocol( &gEfiMpServiceProtocolGuid, NULL, &MpProto);
if (EFI_ERROR(Status)) {
    Print(L"Unable to locate the MpService protocol: %r\n", Status);
    break;
}

// Get Number of Processors and Number of Enabled Processors
Status = MpProto->GetNumberOfProcessors( MpProto, &NumProc, &NumEnabled);
if (EFI_ERROR(Status)) {
    Print(L"Unable to get the number of processors: %r\n", Status);
    break;
}

// Get Processor Health and Location information
Status = MpProto->GetProcessorInfo( MpProto, ProcNum, &Tcb);
if (EFI_ERROR(Status)) {
    Print(L"Unable to get information for proc. %d: %r\n", ProcNum, Status);
}
StartAPs

// Create an Event, required to call StartupThisAP in non-blocking mode
Status = gBS->CreateEvent( TPL_NOTIFY, NULL, NULL, &Event);
if (Status == EFI_SUCCESS)
{
    Print(L"Successful Event creation.\n");

    // Start a Task on the specified Processor.
    Status = MpProto->StartupThisAP( MpProto, Procedure, ProcNum, Event,
                                     TIMEOUT, ProcedureArgument, NULL);
    if (Status == EFI_SUCCESS) {
        Print(L"Task successfully started.\n");
    } else {
        Print(L"Failed to start Task on CPU %d: %r\n", ProcNum, Status);
    }
} else {
    Print(L"Event creation failed: %r\n", Status);
}
Client Task

// Count from 1 to LoopCount
LoopCount = Tcb->MaxCount;   // How high to count.
do {
    ++Counter;                  // Do WORK
    while(Tcb->Ready != 0) {
        // Wait until Result has been Consumed
        CpuPause();              // Hint to CPU that this is a spin loop
    }
    Tcb->Result = Counter;      // Report my results
    Tcb->Ready = 1;             // Signal that Result has been Produced.
} while(--LoopCount);         // Do WORK LoopCount times
  // We have now done all of our work and could exit right now.

// For debugging and paranoia's sake, Send one last "special" value.
while(Tcb->Ready != 0) {
    // Wait until Result has been Consumed.
    CpuPause();              // Hint to CPU that this is a spin loop
}
Tcb->Result = 0xFEEDFACE;    // Indicate that ClientTask is exiting
Tcb->Ready = 1;              // This should be ignored
Root Task

// Retrieve and Display NUMLOOPS values from the Client Task
for(count = NUMLOOPS; count > 0; --count) {
    // Wait until the ClientTask signals data is ready
    while( Tcb->Ready == 0) {
        CpuPause(); // Hint to CPU that we are in a spin-loop
    }

    // Display what we received from the Client
    Print(L"%3d: %4d %4d %4d %4d\n",
          NUMLOOPS-count, Tcb->ProcNum, Tcb->MaxCount,
          Tcb->Result, Tcb->Ready);

    Tcb->Ready = 0; // Tell the Client Task she can run
}

// Give the ClientTask a chance to signal she is done.
while( Tcb->Ready == 0) {
    CpuPause(); // Hint to CPU that we are in a spin-loop
}

// Print the final state of the Tcb.
Print(L"END: %4d %4d %4d %4d\n",
      Tcb->ProcNum, Tcb->MaxCount,
      Tcb->Result, Tcb->Ready);
Summary

• Multi-Processor Programming
  – Platform Control
    > Mp Service Protocol
  – Resource Exclusivity
    > Synchronization Library
  – MP-Safe Libraries
    > Synchronization Library
    > Base Library
  – Interprocess Communication

• The StartCore Application
  – Flexible, Supports Framework and PI Firmware
  – Scalable, Two Cores or Twenty or ???????
Additional resources:

- More web based info:
  - UEFI Community Web Site:  http://www.tianocore.org
  - StartCore sample application:  www.tianocore.org.
  - OS Web links:
    > Link to Microsoft UEFI Support and Requirements:  http://www.microsoft.com/whdc/system/platform/firmware/uefireg.mspx
    > Red hat link:  https://fedoraproject.org/wiki/Features/EFI
  - Whitepaper “Installing UEFI-based Microsoft Windows Vista SP1* (x64) on HP EliteBook and Compaq Notebook PCs” on  www.hp.com