ARGENT

Enhanced Reliability of the Argent Job Scheduler in Tight Core

ARGENT

Background

Argent has focused substantial technical and development resources on assessing and improving the effectiveness of Argent Job Scheduler in heavily constrained environments.

This document describes unexpected behaviors seen in the Windows operating systems and enhancements Argent has introduced in Argent Job Scheduler to compensate for these Windows problems.

Heavily Constrained Environments

Under Windows, any environment is heavily constrained if the amount of real physical memory available to Windows is too low. Argent terms such environments as 'low-memory-high-workload' environments. The amount of real physical memory available to Windows is affected by a number of factors.

For Argent Job Scheduler, an important factor is the number of jobs running at any given time. Another important factor is the exact type and behavior of the jobs.

There is no exact numeric definition of when real physical memory is too low. A general definition is simply that when Windows needs real memory, it can not acquire it.

The amount of real physical memory available to Windows can be viewed in Windows Task Manager and can be retrieved under program control through the Windows API.

Effects on Executing Jobs

In heavily constrained environments, there are effects on executing programs and jobs when real physical memory is insufficient. There are adverse effects on both Windows and non-Windows programs. Argent's focus is on non-Windows programs.

As with all Argent products, Argent Job Scheduler uses a set of self-diagnosing routines to identify issues during execution of programs in Argent products. Using these diagnostic routines, Argent has observed that the effects of heavily constrained environments on executing programs fall into three primary categories:

- Disk File I/O
- Heap management
- C run-time

Disk File I/O

With few exceptions, all disk file management under Windows is fully buffered and managed directly by the operating system.

In our laboratory facilities, Argent's engineers have confirmed and reproduced that in heavily constrained environments disk file buffer write operations can be delayed or postponed for long enough that the effect on an executing program is that the file appears to not exist. Some disk file I/O operations appear to have been 'shed' or 'lost'.

The buffering of disk file write operations affects the directory structure where a file resides as well as the contents of the file.

In heavily constrained environments, the following scenario is possible:

1) A program creates a file. For illustration, call it PROGRAM_A.
2) PROGRAM_A receives confirmation that the new file has been created successfully
3) Disk file I/O management delays the writing of the new file's entry in the parent directory
4) A separate program, call it PROGRAM_B, attempts to read the new file
5) Because of the disk file I/O management delay, PROGRAM_B receives a return code that the file can not be found.
6) Visual inspection confirms that the file does not exist

Recognizing this effect, Argent has engineered solutions in Argent Job Scheduler that preclude the above scenario.

Heap Management

In modern computing environments, any program executing on any platform can allocate memory for variables and internal constructs. Memory can be allocated directly under program control. Memory can also be allocated indirectly, by using operations that cause the operating system to allocate memory on behalf of the program.

In many typical situations, the memory is allocated from a pool made available to an executing program called the 'local heap'. The local heap expands as memory is allocated and contracts as memory is later freed.

Argent has confirmed that in heavily constrained environments, a high number of executing processes can result in scenarios where the heap management software is disrupted. The disruption can become so severe that error message boxes are generated by the operating system explicitly stating the memory from the heap can not be allocated. When the message boxes appear, the program is aborted by the operating system.

Argent has introduced solutions in Argent Job Scheduler that defend against the possibility that memory shortages could result in Job Scheduler programs being aborted by the operating system.

C Run-Time

Many years ago, computer memory was so expensive that programmers and software developers were forced to allocate memory only when absolutely necessary. As memory prices have fallen and memory upgrades have becomes ubiquitous, programmers have become far less conscious of memory requirements for executing programs. This more relaxed approach to memory management can become an issue in heavily constrained environments, especially for programs developed using object-oriented programming (OOP) principles.

In legacy programs, a buffer might be allocated in memory to hold a text value of, say, 128 bytes in length. The programmer knew exactly how much memory had been allocated. In modern object-oriented programs, an executing program might create an object in memory that manages a text value of 128 bytes in length. The object might include resources needed internally by the operating system and methods exposed by the object, in addition to the actual buffer of 128 bytes. It is not always obvious to the programmer exactly how much memory was allocated to store the object.

In heavily constrained environments, executing programs can be affected by errors in the run-time support for some C compilers. Argent has confirmed that, in heavily constrained environments, that the C run-time support available from the Microsoft compiler fails and corrupts the memory of the executing Argent program. Argent observes that this sometimes occurs when a memory requirement of the C run-time support is not satisfied and the shortage is not properly detected.

The effect of this can sometimes be seen on memory that is available to an executing program, but not controlled by the program. In Windows, access to internal constructs exposed by the API is often managed through the notion of a 'handle to memory'. In heavily constrained environments, the handles to the memory allocated by C run-time support can become corrupted, disrupting the execution of programs. Even memory allocated solely to the C run-time support (and not available to executing programs) can become corrupted, disrupting reliable and well-tested software products.

Argent has confirmed that in heavily constrained environments routine compare operations of two integer values that both contain the same value can be made to fail, resulting in logically impossible execution paths.

C Run-Time

Consider the following code fragment:

```
int variable_A;
int variable_B;
variable_A = 123;
variable_B = 123;

if(variable_A     is_not_equal_to     variable_B)
{
log an error and halt if the two values are not equal
}
```

This is almost certainly caused by the Microsoft code writing over the top of the properly running Argent code. Argent has confirmed that in heavily constrained environments, the 'log an error and halt' operation can be can be made to appear, even though the outcome is logically impossible.

Argent has engineered solutions in Argent Job Scheduler that greatly reduce the possibility that the product can be made to fail in this way in heavily constrained environments.

Summary

As part of its long-standing commitment to delivering quality software products, Argent continues to conduct investigations and research into how software can be made to fail and why software failures occur.

Argent's investigations of heavily constrained or 'low-memory-high-workload' environments has led directly to improvements in Argent Job Scheduler that help protect our customers against problems that can occur in disk file I/O, heap management, and C run-time support.