Will the APLNext Supervisor Improve Performance?

Performance improvements due to the incorporation of asynchronous processing in a specific application system are achieved only at the extremes. The extremes intersect when many executions of an in-memory calculation intensive method occur in an application system. Typically these are time series analysis, simulations such as those of weather or chemical reactions, group theory, stochastic modeling and population dynamics. With multi-core processors, the application system programmer may need to appropriately re-configure the application system.

Symmetric multi-processing is supported in APL+Win (v10+) via the APLNext Supervisor. One or more independent threads, each providing a potentially-identical, asynchronously-operating APL+Win instance are spawned by the APLNext Supervisor operated by the 'controlling' application. Based on programmer-design, the controlling application designates the APL+Win function that will be run in each of these independent threads. Programmer re-configuration of a linearly-programmed application system into a multi-processing application involves:

- Identifying those processes which are performed multiple times
- Removing 'side effects' from those processes, such as interactions with other potentially parallel processes, to assure independence
- Ideally modifying those processes to be in-memory operations, eliminating database access or GUI-presentations
- Minimizing the data required by the process arguments as well as the result data returned by the process
- Developing a controlling-application-based protocol for starting such a process, accumulating the results of successful execution of such a process and handling failed execution attempts for such a process.
- Incorporating the APLNext Supervisor methodology into the controlling application.
- Preparing the subordinate APL+Win workspace containing the functions (with explicit results) supporting the application system processes to be executed asynchronously.
- Installing the re-configured application on a machine with multi-core/processor hardware
- Comparing the performance of the linearly-programmed form of the application to that of the multi-processing form of the application

The performance comparison element of the re-configuration will aggregate the benefits of parallel processing with the additional costs of argument and result data marshalling between the controlling application and the threads. Generally symmetric multi-processing yields benefits only in cases of highly-repetitive, in-memory algorithms without 'side effects' that process very large arrays.

The choice of the controlling application can also have a significant impact on performance. Whereas the independent threads spawned by the APLNext Supervisor are designed to run APL+Win functions, the controlling application can be any Win32-based language, such as APL+Win or VB6 or any .Net language such as C#, VB.Net or VisualAPL. Selecting a .Net language for the controlling application provides the benefit of an environment designed for multi-threading, so that results from the asynchronous APL+Win threads can be absorbed by the controlling application more efficiently.

An alternative to programmer-controlled decisions about which application system processes should be performed asynchronously, is automatic multi-processing at the APL+Win array-based operator level. Automatic multi-processing involves incorporating into the language a cost-benefit analysis to determine if an array-based operation could benefit from parallel processing. As of APL+Win v10, automatic multi-processing has not been implemented because the performance cost to the overall application of such operator-level analysis by the interpreter is not insignificant. Adding interpreter analysis to language-level operators to detect the relatively rare conditions where multi-processing would yield benefits burdens the performance for the vast majority of operator executions.

As of APL+Win v10, the APLNext Supervisor provides easy access to multi-threading based on application system programmer intervention to identify the processes that would benefit from asynchronous execution. The architecture of the APLNext Supervisor is such that it does not degrade the performance of an APL+Win-based application system that does not employ the APLNext Supervisor technology, as automatic multi-processing might.