IBM'S ZIIP FOR MAINFRAME EXPLAINED

As a follow-up to Phil Grainger's post about zIIP offload, I thought it was worth spending a few moments to focus on the characteristics of code that can run on a zIIP as well as code that could suffer minimum performance loss if it was rewritten to run on a zIIP rather than a general purpose processor (GP).

IBM introduced the zIIP (system Z Integrated Information Processor) in 2006. IBM licenses zIIP hardware for a fraction of the corresponding charges for GPs. More importantly to you, zIIP processors are not counted in the basis for software charges, so you can see major savings by shifting workload from GPs to zIIPs. In addition, zIIP processors are run "at full speed," so you can actually get more processing done in less time if your GPs are not full capacity models.

So why not simply replace every GP with a zIIP and save money on both hardware and software?

First, IBM has placed a restriction on zIIP usage. Originally, there could be no more than one zIIP per GP in a CEC, but some models now allow two zIIPs per GP. Second, IBM's license agreement places restrictions on the kind of code that is eligible to run on a zIIP; the code must run in a z/OS enclave under the control of an SRB (service request block).

The use of an enclave means that the work is dispatched by z/OS Workload Manager (WLM). Code run from the enclave can be handled by a common set of WLM controls and policies because the code has a common set of characteristics. Scheduling an SRB instead of attaching a task control block (TCB) implies further restrictions; the most pertinent of these is that no SVC (except ABEND) can be issued, which prevents the use of many z/OS services commonly invoked by application programs. This has been misunderstood as "you cannot do I/O on a zIIP," which is not true. However, it is true that QSAM and BSAM, which many application programs use to perform sequential file I/O, cannot be invoked from SRB code because they issue SVCs.

For completeness, we should mention "zAAP on zIIP," by which IBM allows processes that are zAAP eligible to run on a zIIP in environments where there are no zAAPs. Very little legacy code fits this profile, but we are seeing more Db2 installations that are converting legacy COBOL applications to Java because programmers are easier to find than ones with COBOL skills. A side effect of this move is to render legacy applications to be zIIP eligible after they are rewritten into Java.

So why aren't your mainframe vendors zIIP enabling all of the code in all their products? The answer gets back to the second restriction mentioned above – to be zIIP eligible, code must run under an SRB and therefore cannot invoke many z/OS services. Most existing product code was written to run in TCB mode. Historically, SRB mode code was used only where it was considered a cost effective way to do cross memory processing.

Let's look at a hypothetical piece of code (part of a software product) that runs under a TCB today and makes no SVC calls. To make the code zIIP eligible, we must do something like the following each time we execute the code:

1. Set up the SRB in memory to point to the code to be executed.
2. Issue the IEAMSCHED macro to schedule the SRB for execution.
3. The originally executing TCB code must wait for the SRB code to execute.
4. When the SRB code completes, it must signal the TCB code to resume before terminating.

Alternatively, we could start the SRB once, then loop and wait in the SRB code instead of scheduling a new SRB each time the function is to be performed.

The overhead incurred in moving from TCB to SRB mode and back will incur a CPU cost. Vendors weigh the advantages of making code zIIP eligible with the CPU costs to determine what code will be most efficient for users.

Vendors should make their products zIIP eligible where feasible. They should focus on programs that are CPU intensive, don't contain the z/OS services calls that cannot be made from an SRB, and don't require much I/O. The programs must be executed often enough that the CPU savings on your GPs will be substantial.

While it would be nice if we could enable all of the code in every one of your z/OS software products to run on zIIP instead of GPs, that is unrealistic.

One last thing is that the license that allows third party vendors to zIIP-enable their code specifically prevents use of the license to zIIP enable OTHER code to run on a zIIP. So a vendor can zIIP-enable code they have written, but they could not write a tool to zIIP-enable a customer's code.

I hope the information in this article will provide you with the information you need to discuss zIIP exploitation with your vendors.