What is Batch Processing?

#4 IN A SERIES

"**Batch processing**" has a long history and is very pervasive and very important in most Enterprise Computing[1] environments. However, it is still somewhat of a foreign concept to those that have not used an enterprise server[2]. A personal computer and distributed computers do not have a direct counterpart to batch processing. This article explores "batch processing" and its characteristics.

Batch processing was so named because in the early days of commercial computing, programmers would bring their programs (in the form of punched cards) to the computer to be run. The collection of programs from multiple programmers would be stacked together and run through a card reader so that the computer could serially process the "batch" of programs. The programs would run unattended and do whatever they were designed to do. The programmers that submitted them would eventually see the output from their program, usually in the form of a print out. Once the processing of the batch of programs was complete, another batch would be collected and the cycle would repeat.

Computers have evolved substantially from those early days, but batch processing still continues to serve a useful purpose for businesses that have processing needs that match its characteristics. From the brief description above we can see that batch processing involves:

- the unattended running of a program or programs
- the collection of program input data collected over some period of time
- the output from running the programs is not needed immediately

If you are familiar with transaction processing[3] and its user submitted input and sub-second response times for output, these characteristics may seem foreign to you. But actually, there are several business processes that exactly match these characteristics. A classic example is business billing. All businesses bill for their products or services. Think of an electric utility or a credit card provider. Each sends a customer bill monthly that must be printed and mailed (or e-mailed). It would be a waste of a person's time to manually enter a transaction that generated each bill, one at a time. Instead, a program can be started that reads all the data associated with each customer's activity during the time period of interest (e.g. a month) and then produce a bill as output. The customer input (e.g. utility usage) is collected during the billing period. The input to the program may be sorted prior to processing so as to produce the customer bills in a particular order (perhaps by zip code if they are to be mailed). There is no need for human intervention or monitoring of this bill generation processing. And, immediate results or output is not crucial. If it takes several hours or days to generate the bills for all of the customers, the billing program can simply be started prior to the time the bills need to be ready.

There are many other examples of business processes that can be effectively accomplished using batch processing. These include:

- computer programs that need to go through a "compile, bind, execute" cycle many times during development and test of a program
- the printing of customized "junk" mail advertising
- the conversion of a large number of files from one format to another
- the movement of a large number of files from one device to another to prepare data for data mining
- the entry of orders for a business' products that have been mailed in
- the printing of standard reports or statements

While there is no direct counterpart to batch processing on a personal computer, there are some

[1] See ECI No. 1
[2] See ECI No. 5
[3] See ECI No. 3

common personal computer activities that you may be familiar with that exhibit some of the characteristics of batch processing. Virus scanning and the wholesale processing of graphic images are two examples. In the case of a virus scanning program, the processing can be scheduled to run periodically and it runs unattended, processing a large number of files. The virus scanner is looking for a virus in the data or programs that have been collected over some time period (i.e. since its last scan). The program output (the virus scan report) is not needed immediately and in fact takes a considerable time to complete. And, other "more important" work (your use of a word processor for example) can proceed while the virus scanner runs in the "background".

If you are familiar with the UNIX operating system, processes run using "at" or "cron" are somewhat analogous to batch processing on an enterprise server.

The unit of work in a batch processing environment on an enterprise server is the JOB – one or more programs that are run to accomplish some business purpose. Each portion of the JOB that runs a program is called a JOB "step". If some input data needs to be sorted by a sort program before another program is run to print bills, a two-step JOB would be defined in which the first step runs the sort program and the second step runs the program that prints the bills (using the sorted output file from the first step as input to the second step).

A JOB is defined using Job Control Language (JCL) statements. The JCL statements are used to describe certain attributes of the JOB. For example, the JCL statements describe what datasets are needed as input by the programs that will be run, what output will be generated (disk files, tape files, printed output, etc.), the JOB priority (although this has become less important with the advent of a sophisticated workload manager), and other attributes.

Batch JOBs can be run on an "as needed" basis or even more likely batch JOBs can be scheduled to run at a particular time since their output is usually tied to a particular business schedule (e.g. end of day, end of month, end of quarter, etc.). JOBs can be manually submitted by a person to run at a particular time, but most enterprises use JOB scheduling software that is designed to automatically submit JOBs based on a schedule. A large enterprise may have thousands of JOBs defined to run on their enterprise server. Some of these JOBs may be totally independent of each other and some of the JOBs may be related in terms of achieving a business objective. An example of a relationship is one JOB creating an output file that is used as input by another JOB. Consequently, a JOB may be part of a larger work effort (a "network" of JOBs) that can also be managed by a JOB scheduling program. In this case, the JOB scheduling program can manage any dependencies between the JOBs. For example, in addition to scheduling the first JOB in the network to run at a certain time, it can also recognize if a failure occurs and avoid starting a JOB that is dependent on the successful completion of a predecessor JOB.

Besides directly supporting many business processes, batch processing provides benefits from an Information Technology perspective as well.

- Since JOBs run unattended, it reduces the possibility that computing resources are tied up, but not actually being used.
- Since batch work can be scheduled, it can be used to shift the time for processing the batch work to a time when the compute resources are not busy. Historically, this would be late at night (e.g. after midnight). Some businesses would even shut down the on line transaction oriented workloads and use the computer resources to run batch workloads exclusively. This resulted in a "batch window" during which all the batch work for the day had to be completed before the on line work was restarted the next day. With more businesses becoming global in nature and with the advent of the internet, this practice is becoming less

prevalent. And time shifting of batch workload is becoming less of a benefit as compute resources are routinely being used around the clock.

- A collection of batch work allows for the effective sharing of computer resources among many users and programs with the operating system managing any contention.
- And finally, batch work can be thought of as “background” or “filler” work that uses compute resources when they are not needed by more important work (e.g. on line transaction processing). By helping to maintain a high overall rate of compute resource utilization, it helps make better use of the enterprise server.

So, while batch processing lacks the immediacy of results provided by transaction processing, what it lacks in immediacy, it more than makes up for in efficiency, and therefore, cost effectiveness.

We will explore other aspects of Enterprise Computing in subsequent articles.