WHAT IS A BATCH JOB?

The term ‘batch job’ originated in the days of mainframe computers when users entered programs onto punch cards. The user would give a batch of these programmed cards to the system operator, who would then feed them into the computer.

The definition and usage of batching has undergone changes in recent years, and we are finding new ways to perform these tasks. At its core, though, background processing will never die. Whether for routine organizational processes, like payroll, or to keep servers from being tied up during the workday, batch jobs will be essential for years to come.

With this in mind, we have put together this introduction to batching and batch jobs, with a focus on how they’ll maintain their value for enterprises and employee workflows.

How does batching work?

In the simplest terms, a batch job is a scheduled program that is assigned to run on a computer without further user interaction. Batch jobs are often queued up during working hours, then executed during the evening or weekend when the computer is idle.

Once the batch job is submitted, the job enters into a queue where it waits until the system is ready to process the job. If the job queue contains many jobs waiting to be processed, the system processes the jobs either in chronological order or by priority.

Batch jobs are frequently used to automate tasks that need to be performed on a regular basis, like payroll, but don’t necessarily need to occur during the day or have an employee interacted with the system. Jobs that happen on a regular basis are incorporated into batch schedules.

In some computer systems, batch jobs run in the background while interactive programs run in the foreground, giving these interactive programs priority over batch programs. This makes sense because batch jobs are provided with more allocated memory than those performed in the foreground. They are used to process high volumes of data that would normally consume long-term memory if run in the foreground.

The most valuable benefits of batch jobs include:

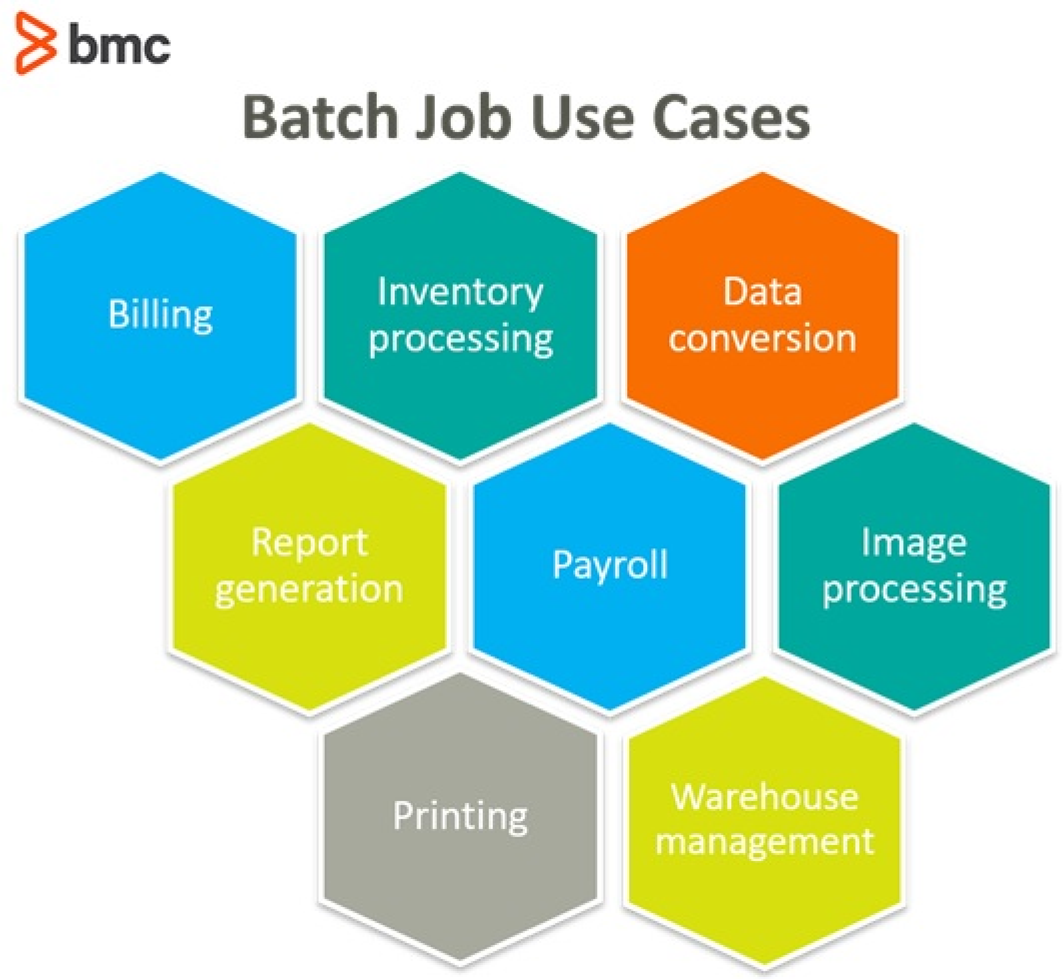
- Large programs can utilize more dedicated servers when the work processes are in night mode.
- With fewer users or employees online during off hours, the performance will be faster and more efficient at night. During the day it could be restricted to fewer servers and numbers needed.
- Employees are freed up to focus on less repeatable, more creative tasks.

Batch job use cases

Traditional batch jobs are still highly relevant activities in almost every business computing environment to this day despite the advances in modern technologies.

A telephone billing application is a perfect example of a batch job. First, the application reads the phone call records from the enterprise information system. Then, the app generates a monthly bill for each customer account. This type of task can run without anyone interfering with it.

Other common uses of batch jobs include:



Steps in a batch job

A step is an independent and sequential phase of a batch job. Batch jobs contain both chunk-oriented steps and task-oriented steps.

Chunk-oriented steps

Chunk-oriented steps process data in three parts:

1. The data is read and processed one item at a time from a specific data source, such as files in a directory or entries on a database.
2. Business processing manipulates each item using the business logic defined by the application, like formatting or filtering.
3. The results are grouped into a chunk, which are stored once the chunk reaches a configurable size.

This type of processing typically makes storing results more efficient and helps facilitate the limits of the transaction.

Task-oriented steps

Chunk steps often process large amounts of data and in turn, can take a long time to complete. Utilizing batch frameworks, these chunk steps are able to then bookmark their progress using checkpoints. Using the input retrieval and output writing parts of a chunk step, the current position is saved after it is processed. Should a certain chunk step be interrupted, it can be restarted from the last checkpoint.

Parallel processing

For batch jobs that require large amounts of data to be processed or for computations involving a very large number of independent computations, parallel processing may be beneficial, reducing the total number of jobs submitted.

Parallel processing is typically most beneficial in two scenarios:

- Steps that don't depend on each other and can run on different threads.
- Chunk-oriented steps where the processing of each item doesn't depend on the results of the previous one.

Batch frameworks provide mechanisms for developers to define groups of specific independent steps, splitting chunk steps into parts that can then run in parallel.

Decision elements

In addition to steps, batch jobs can also contain decision elements. Decision elements utilize the exit status of the previous step to determine the next step or to terminate the batch job altogether. Decision elements set the status of the batch job when terminating it, noting if it was terminated successfully, it was interrupted, or it failed.

BMC for batching and workflow orchestration

Batch jobs have come a long ways since the days of punch cards, but one thing stays the same: automatically processing necessary tasks and functions reduces stress on busy servers while improving system efficiency and productivity.

A leader in modern batch processing and workflow orchestration, BMC Blogs offer a variety of additional articles. Get started with these:

- Modern Batch. To Batch or Not to Batch? There's no question!
- Modern Batch Processing: A Thing of the Past or Essential Discipline?
- Putting Workflow Orchestration to Work for You
- Open Source and Workload Automation
- A Dynamic Duo: Workload Automation and Robotic Process Automation
- Control-M Reviews and User Ratings: The Definitive List

Control-M makes it easy to define, schedule, manage and monitor workflows, ensuring visibility and reliability, and improving SLAs. Learn more or start a free trial of Control-M.