IBM BatchPipes OS/390 Introduction

GA22-7459-00

IBM BatchPipes OS/390 Introduction

GA22-7459-00

Third Edition, September 1995

This is a major revision of, and obsoletes, GC28-1214-01.

This edition applies to Release 2 of IBM BatchPipes/MVS (5655-065) and to all subsequent releases and modifications until otherwise indicated in new editions or technical newsletters. See the Summary of Changes for the changes made to this manual. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

IBM welcomes your comments. A form for readers' comments is provided at the back of this publication, or you may address your comments to the following address:

International Business Machines Corporation
Department 55JA, Mail Station P384
522 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 914+432-9405
FAX (Other Countries):
Your International Access Code +1+914+432-9405

IBMLink (United States customers only): KGNVMC(D58PUBS)
IBM Mail Exchange: USIB2NZL at IBMMAIL
Internet: d58pubs@vnet.ibm.com

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

- Title and order number of this book
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Introduction to BatchPipes

BatchPipes OS/390 offers a way to connect jobs so that data from one job can move through processor storage to another job without going to DASD or tape. It addresses a growing problem that faces installations with batch workloads: insufficient time to complete that work. Given a batch jobstream with a data flow of certain characteristics, BatchPipes can dramatically shorten the elapsed time of the jobstream. The jobs can run faster and process larger volumes of data in the available batch window; your processors are freed to run more work — perhaps extending the period of time that interactive applications are available or supporting your company's work in another time zone. In short, you might get more work from your processors.

BatchPipes, running on MVS/ESA[1]:

- Allows two or more jobs that formerly ran serially to run concurrently.
- Reduces the number of physical I/O operations by transferring data through processor storage rather than transferring data to and from DASD or tape
- May reduce tape mounts and use of DASD.

What difference does parallel processing make to my batch workload? Large tightly-coupled processing systems running with MVS can have many jobs in various stages of processing at one time. Batch applications have not traditionally taken advantage of this multiple processing capability. Rather, they often consist of multiple jobs that run one after another. With BatchPipes, the processor can run two or more jobs in a jobstream at one time. Jobs that once ran sequentially can now run in parallel because data records or blocks are available to the next job immediately after they are written. That is, the whole sequential file does not have to be written and then closed before the next job can access the file.

What difference does keeping data in the processor make to my batch workload? When data moves from one job to another through processor storage, the transfers take place in microseconds as compared to the milliseconds required to transfer data to and from tape or DASD. Keeping data in processor storage reduces the number of physical I/O operations, causes less I/O contention, and frees the device that holds the intermediate data set. An additional benefit of transferring data through processor storage is a reduction in the mounting and managing of tapes for applications using BatchPipes.

An Example of How BatchPipes Changes Traditional Batch Processing

Most installations have batch jobstreams that use intermediate data sets such that output from one process (a job or step) is written to DASD or tape. This data is then available to be read by a second process immediately after the data set closes or the first process ends. Figure 1 on page 2 shows the traditional job-to-job data flow:

[1] MVS/ESA is a trademark of the International Business Machines Corporation.

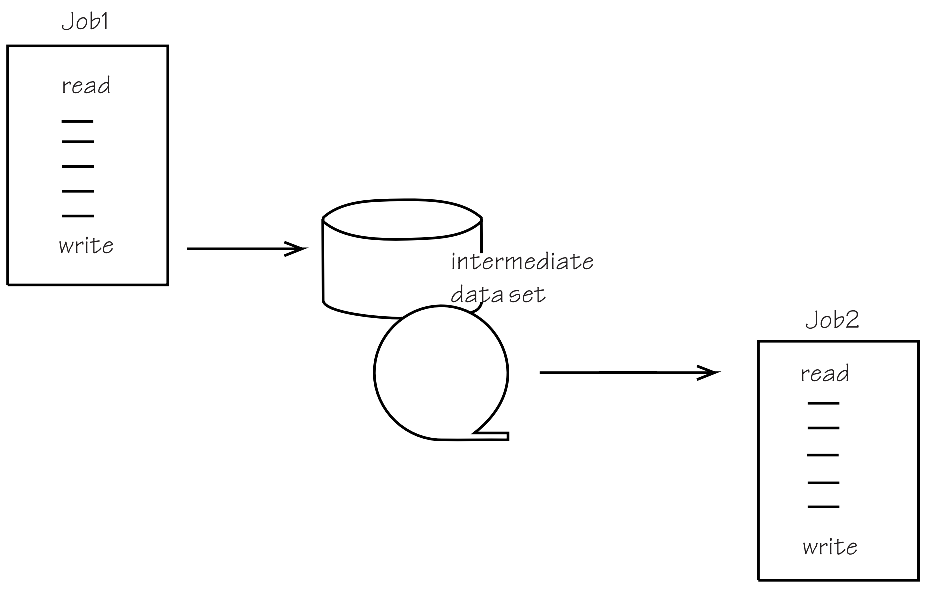

Figure 1. Two Jobs in a Traditional Batch Jobstream

In the figure, Job1 writes data to an I/O device. When all the records have been written and the data set is closed, Job2 starts. Job2 reads the data from the device. After Job2 finishes processing all the records, the data set is no longer required and can be deleted. In effect, the transfer of data from one job to the other is at a data set level.

Compare the processing of the two jobs in Figure 1 with Figure 2, which shows those same two jobs using BatchPipes. Notice that external storage devices are absent, replaced by a processor storage buffer holding a small portion of the intermediate data set.

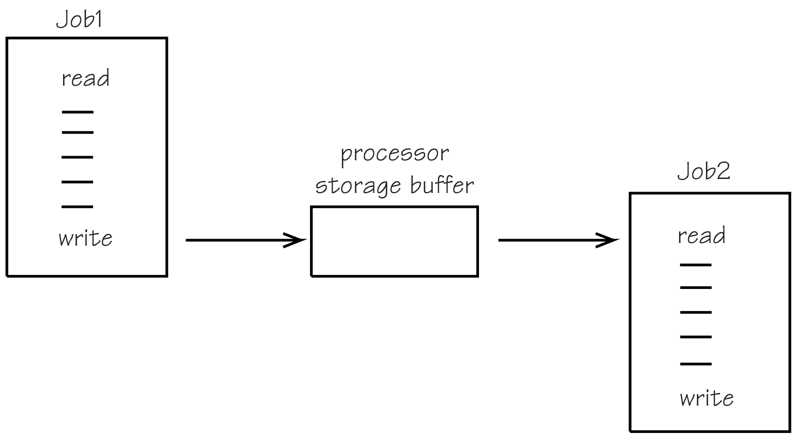

Figure 2. Two Jobs in a Batch Jobstream using BatchPipes

In Figure 2, Job1, a **writer** to the processor storage buffer, and Job2 , a **reader** from that buffer, run concurrently. Job2 can obtain data from the processor storage buffer as soon as Job1 writes the first block. Output from Job1 becomes immediately available as input to Job2. You can think of the data as “flowing” from Job1 to Job2. The processor storage buffer is called, in BatchPipes terms, a **pipe**, through which data flows, always in the same direction, from a writer job to a reader job. Writer-->pipe-->reader are known as a **pipeline**.

To understand the elapsed time savings, compare timelines of traditional job-to-job processing and BatchPipes job-to-job processing. The timeline for the traditional processing of Job1 and Job2 might look like this:

```
<------Job1----><--------------Job2------->
___________________________________________
'  '  '  '  '  '  '  '  '  '  '  '  '  '  '
0     2     4     6     8     10     12    14
```

Figure 3. Timeline Showing a Traditional Two-Job Jobstream

With BatchPipes, the two jobs can run concurrently in **less** time than Job2 (the longer-running job) formerly needed. Elapsed time savings come from concurrent processing of the two jobs, from elimination of **I/O wait times** (time intervals during which a job waits for I/O transfers to and from DASD or tape), and from reduction of I/O transfer times between jobs.

The timeline for the two jobs using BatchPipes might look like this:

```
<-------Job1----------->
 <-----------Job2------->
___________________________________________
'  '  '  '  '  '  '  '  '  '  '  '  '  '  '
0     2     4     6     8     10     12    14
```

Figure 4. Timeline Showing Two Jobs Using BatchPipes

Running the two jobs concurrently demonstrates a form of **parallelism**, the simultaneous processing of many tasks (or jobs) within one application. Figure 4 shows two jobs running concurrently in a multiprocessing environment. Without BatchPipes, the jobs would not run concurrently.

Resources Required by Jobs Using BatchPipes

BatchPipes runs as a subsystem installed on MVS. The required software is MVS/ESA SP Version 5.0 or later, running with the levels of JES and DFP that support the release. The hardware must be able to run the software.

As you plan to use BatchPipes, remember that the writer and reader run concurrently; therefore, all resources for those jobs must be available for the shorter time span. The important resources you need to consider are initiators, tape and DASD, processor storage, and the processor itself.

- Initiators

  Initiators must be available for all the jobs in a pipeline. Until all jobs have initiators, data cannot flow through the pipeline.

- Tape and DASD

  BatchPipes reduces the need for physical storage for temporary data sets. However, you still need enough devices to simultaneously hold all writers' and readers' data that is outside the control of BatchPipes.

- Processor storage

  Each job has its own need for processor storage. When multiple jobs run concurrently, the combined storage for all jobs must be available. In addition, each pipe requires enough storage to hold a number of blocks of the data set as the data flows through the pipe.

- Processor

  Each job has its own need for the processor. When multiple jobs run concurrently, the processor works on behalf of them all. In addition, to reduce the cost of I/O to external devices, BatchPipes moves data through processor storage. This action might increase a job's processor time, as reported by system management facilities (SMF).

Choosing Candidates for BatchPipes

BatchPipes offers large improvements to some applications, but not to all. This section describes two approaches for identifying jobs to use BatchPipes; which approach you use is determined, to some extent, by how much work your application developers are willing to do to gain the BatchPipes improvements.

- The first approach is to examine your batch jobs and identify those sets of jobs that would be **good candidates** — those sets of jobs that can use BatchPipes after a minimal JCL change.
- The second approach is to identify those applications that are critical to your business and estimate the work required to change the application to use BatchPipes.

  To effectively use BatchPipes to speed up a given jobstream, you might need to reduce the resource contention caused by the other jobs in the system. If other factors are equal, when it comes to a choice between candidates for BatchPipes, the jobstream that is more important to your business is generally the better candidate.

Whichever approach you use, or if you use a combination of the two, you need to know what makes good candidates for BatchPipes:

- A job or jobstep depends on completion of another job or jobstep for its data; where one job or jobstep writes a data set and then another job or jobstep reads the same data set.
- The jobs process one record or block at a time; that is, they read a block, process it, write a block, and repeat the pattern until all blocks are processed.
- The jobs use sequential QSAM, BSAM, or GSAM-BSAM to access the data set. (Note that BatchPipes does not support checkpoint and restart requests.)
- The data set is intermediate; that is, the data set should not exist after the reader uses the data. This is not a restriction however, as you can see in “Using a Filter to Make Two Copies of a Data Set” on page 11.
- The intermediate data set has a record format of fixed or variable, which can be blocked and can use machine print control characters
- Writer and reader cannot have resource conflicts; for example, writer and reader cannot both require exclusive use of a data set.
- The two jobs can run on the same MVS system.

In following the first approach, examine your batch work and identify sets of jobs that meet the basic criteria. Next, analyze critical paths of those jobs that meet the basic criteria. This analysis requires knowledge of the elapsed time of jobs and job dependencies:

- Elapsed time of jobs

  You might choose to start your analysis by looking at those jobs that run for long periods of time. But don't overlook those jobs that run for short lengths of time. Speeding up several small jobs can improve the elapsed time of the entire jobstream.

- Job dependencies

  If a job or jobstep depends on completion of an external event, that job or jobstep might not meet the BatchPipes scheduling requirement that the reader and writer jobs run concurrently. Analyze external dependencies by studying job log data that identifies gaps that occur between when a writer job ends and a reader job starts. This gap can indicate that the reader job is:

  – Waiting for a data set to be closed
  – Waiting for delivery of a tape from another location
  – Waiting for another job to complete
  – Dependent on a time-of-day event.

  When writer and reader jobs or job steps must both wait for external events, they may not be good candidates for BatchPipes.

You might have the results of the BatchPipes analysis tool on which to base your decision. That tool, run by IBM, analyzes customers' SMF data and produces a report on the jobs that use sequential QSAM and BSAM intermediate data sets. It identifies jobs that are likely to be good candidates for BatchPipes and estimates elapsed time savings. Additionally, IBM offers an integrated services offering (ISO) to give you a more detailed analysis of job candidates and QuickStart Consulting Services to get you started using BatchPipes.

Ask for our help ...

For more information about the analysis tool, the ISO, and the QuickStart Consulting Services:
Call 1-800-IBM-7890 or send a note to USIB594L at IBMMAIL

Comparing BatchPipes with Other IBM Solutions for Batch Applications

BatchPipes is one of several software and hardware functions that IBM offers to improve the elapsed time of batch jobs that run on MVS. You might be familiar with:

- VIO to expanded storage
- Hiperbatch
- Batch Local Shared Resources (BLSR)
- Sequential Data Striping.

To answer questions you might have about how BatchPipes compares with these functions, this section looks at whether these functions benefit the same type of batch jobstream that BatchPipes benefits, a jobstream where:

- The jobs use sequential QSAM, BSAM, or GSAM.

- Output from one job or jobstep is input to a second job or jobstep
- The data set is intermediate. (Note that BatchPipes provides several ways to customize the data flow in case you need a permanent copy of data for backup or recovery.)

Both **VIO to expanded storage** and BatchPipes target similar batch environments. They eliminate the I/O wait times that occur when a job waits for I/O transfers to and from storage devices; they handle data that is intermediate. VIO keeps **all** the data in expanded or auxiliary storage from the time the data is created until the entire data set is deleted; BatchPipes keeps only a small portion of the data set in virtual storage at one time. Therefore, there is no limit to the amount of data that BatchPipes can transfer.

VIO supports both random and sequential I/O; BatchPipes supports only sequential I/O.

VIO improves performance for jobs that use data sets that are relatively small. If the job requires large quantities of data (for example millions of records), VIO is not suitable because expanded or auxiliary storage will be exhausted, or because the storage will be paged (eventually to DASD) and performance could actually decrease.

Because BatchPipes allows writer and reader jobs to run concurrently, elapsed times generally improve more with BatchPipes than with VIO.

VIO requires the use of expanded storage; BatchPipes does not.

Hiperbatch targets a different type of batch application from BatchPipes. With hiperbatch, the target application has many readers simultaneously accessing data that one writer job created. The shared data **all** resides in processor storage and on DASD; the data persists after being read, subject only to reuse of hiperbatch buffers by other hiperbatch activity. In contrast, BatchPipes holds very little of the data in processor storage at one time, and the output from the writer job is temporary.

With hiperbatch, readers run simultaneously with each other, but not with the writer; with BatchPipes, writer and reader jobs run simultaneously.

With hiperbatch, there's always the chance the reader might read from DASD, rather than processor storage; with BatchPipes, the reader always finds the data in processor storage.

BLSR also targets a different type of batch application than BatchPipes. With BLSR, the target application uses direct or keyed VSAM in a non-sequential access pattern. In contrast, BatchPipes supports QSAM and BSAM in a completely sequential access pattern. Further, for data integrity reasons, BLSR writers and readers must run serially. BatchPipes writers and readers must run concurrently.

BLSR is not generally suitable for sequential access patterns (because the performance could actually decrease) and BatchPipes is not suitable for random access patterns.

Sequential Data Striping benefits batch jobs that spend much of their elapsed time sequentially reading or writing large DASD files. When a data set is created, Sequential Data Striping allocates the data set on as many as 16 volumes. Jobs that later read the data have their read requests processed in parallel from those devices; Sequential Data Striping automatically "reassembles" the records as the data is transferred from DASD. Once all the records are reassembled, Sequential Data Striping returns control to the job.

With Sequential Data Striping, the data is kept on DASD; with BatchPipes, the data temporarily resides in processor storage. Sequential Data Striping reduces I/O operations and I/O wait times; BatchPipes eliminates them.

Preparing to Use BatchPipes

BatchPipes is not part of MVS itself; it is a subsystem, running as a started task under MVS. Installing, initializing, and starting BatchPipes is similar to getting any SMP-installed program product up and running. For your initial release of BatchPipes, a system programmer installs the BatchPipes code and initializes one or more instances of BatchPipes subsystems (using parmlib members to define the subsystem to MVS), then IPLs MVS. With proper planning, subsequent installations of BatchPipes do not require an IPL.

Once BatchPipes is installed and candidate jobs examined and chosen, using BatchPipes with your jobs requires JCL and scheduling changes.

Data Set DD Statement Changes: The DD statements that describe the *output data* from the writer and the *input data* to the reader must tell the system to use BatchPipes, rather than the I/O subsystem, to transfer the data. The specific parameter on the DD statement is:

SUBSYS=*subsysname*

where *subsysname* identifies the BatchPipes subsystem that is to control the data flow.

For example, to tell the system that the writer of the data set named ACCTRD.TEMP is using the BatchPipes subsystem named *BP01* to control its I/O requests, the DD statement looks like the following:

```
//SYSUT2 DD DSN=ACCTRD.TEMP,SUBSYS=BP01,
//          LRECL=80,RECFM=FB
```

The reader job would have a similar JCL specification, including the SUBSYS=BP01 parameter.

Scheduling Changes: You might need to change the scheduling of the jobs so that they can run concurrently. This might mean:

- Changing input to job scheduling packages that you are using, such as Operations, Planning & Control/ESA (OPC/ESA)
- Evaluating the use of initiators to reflect the added job or jobs that are competing for the processor's use.

Application JCL Changes: BatchPipes transfers data from one job to another. To use BatchPipes to transfer data between two or more jobsteps in the same job, JCL statements must be changed to turn the jobsteps into separate jobs.

Another JCL change that might increase the benefit of BatchPipes is described in "Running BatchPipes with Multiple Copies of a Job" on page 10.

Application Code Changes: BatchPipes requires no changes to applications' normal I/O declares, macros, or routines; it intercepts application requests for I/O and passes the data through processor storage.

Running BatchPipes

BatchPipes is controlled through MVS or BatchPipes commands. The MVS START command starts the subsystem.

The MVS STOP (or BatchPipes STOP) command shuts down the BatchPipes subsystem, allowing all writers and readers already using the subsystem to complete their work, but preventing new writers and readers when using the subsystem.

The MVS CANCEL (or BatchPipes CANCEL) command shuts down the BatchPipes subsystem and attempts to cancel the jobs using the subsystem.

Tracking BatchPipes Activity: The BatchPipes STATUS command describes BatchPipes activities. An operator's screen can show:

- The activity of a specific job or jobs
- The activity of a specific pipe or pipes
- The status of a BatchPipes subsystem
- Job-pipe-job relationships.

This information can help computer personnel make changes to the pipeline or change the number of writer or reader jobs, as described in “Running BatchPipes with Multiple Copies of a Job” on page 10.

The BatchPipes monitor, provided through an ISPF interface, allows TSO/E end users to obtain information about:

- A particular subsystem
- One or more jobs
- One or more pipes
- Thresholds and other subsystem settings
- Job-pipe-job relationships.

Tracking the Use of Resources: SMF records allow you to track the use of the BatchPipes subsystem and to measure its effectiveness. SMF collects information about all pipes and jobs associated with the subsystem. Analyzing the records can show:

- The number of pipes defined
- The jobs attached to each pipe
- The number of read and write requests to a pipe by a job
- The amount of data a job has transferred through a pipe
- The total amount of data transferred by a subsystem.

Managing the Workload in the BatchPipes Environment

Managing the workload in the BatchPipes environment requires your computer personnel to understand new dependencies that jobs using pipes have on each other. The best way to describe the BatchPipes environment is to build on an example used earlier in the book. Consider a hypothetical non-BatchPipes jobstream that consists of JobA, JobB, and JobC. JobA runs to completion in 8 hours. All output from JobA must reside on tape before JobB starts. JobB takes 6 hours to complete and write its output to tape; JobC takes 8 hours. A timeline of the three jobs running along with other work might look like this:

```
<------------------------  Other work  -------------------------->

<--------JobA-----------><------JobB------><---------JobC--------->
__________________________________________________________________
'  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '
0     2     4     6     8    10     12    14    16    18    20    22
```

Figure 5. Timeline of Three Jobs in a Traditional Batch Environment

With BatchPipes controlling the flow of data, the three jobs run concurrently. JobB can read data as soon as JobA writes it; JobC can read data as soon as JobB writes it. The data flows through the pipeline that consists of JobA-pipe-JobB-pipe-JobC. The following timeline describes the concurrent processing of the jobs.

```
<--------JobA---------->
 <----------JobB-------->
  <-----------JobC------->
__________________________ . . .
'  '  '  '  '  '  '  '  '
0     2     4     6     8
```

Figure 6. Timeline of the Three Jobs Running Using BatchPipes

Now, the jobs have new **and immediate** dependencies on other jobs in the pipeline. If JobC has to wait for an operator to mount a tape, JobA and JobB might also wait. If JobC ends (or “times out”) because of that delay, JobA and JobB might also end.

This example also shows how running BatchPipes changes the use of resources. Assuming that other work is in the system **and** you do not reschedule that workload, having the three concurrently-running jobs means:

- The other work that runs during the first 8 hours of this elapsed time period has new competition for use of the processor and processor storage
- The other work could slow down the jobs in the pipeline, depending on their use of resources
- Resources for the three jobs are needed over an eight- or nine-hour period, rather than the original 22-hour period.

You can schedule the BatchPipes jobs in a known batch window and schedule the other jobs at a different time, as Figure 7 suggests.

```
<--------JobA--------->|         This time is now                  |
 <----------JobB------->|     available for other work.            |
  <-----------JobC------>|                                         |
___________________________________________________________________
 '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '
0     2     4     6     8    10     12    14    16    18    20    22
```

Figure 7. Timeline of the Three Jobs and the Rescheduled Work

Although the example is realistic, keep in mind that not all jobs can run in parallel as JobA, JobB, and JobC did. Also, elapsed time improvements brought by BatchPipes vary according to the characteristics of the jobs.

Running BatchPipes with Multiple Copies of a Job

So far, we have concentrated on writer/reader pairs where one writer passes data to one reader. BatchPipes can transfer data from more than one copy of the writer to more than one copy of the reader. To understand why you might want to use BatchPipes with multiple writers or readers, consider what happens when a writer makes write requests faster than the reader makes read requests, or vice versa. In both cases, one of the jobs must wait for the other. In fact, it is unlikely that the writing and reading would be exactly “in sync.” For example, look at Figure 2 on page 2. Suppose Job1 is able to write data twice as fast as Job2 can read. This imbalance leaves Job1 waiting much of the time for Job2 to catch up.

Figure 8 on page 11, in contrast, shows two copies of the reader reading from the buffer, processing the output of Job1. Job2 and Job2A take turns removing records from the pipe. Once read, a record ceases to exist in the pipe; the next read request is for the record that follows. This kind of read operation is known as a **destructive read**. Because ***each job reads only a portion of the data set***, output from the two jobs might need to be merged in a later process.

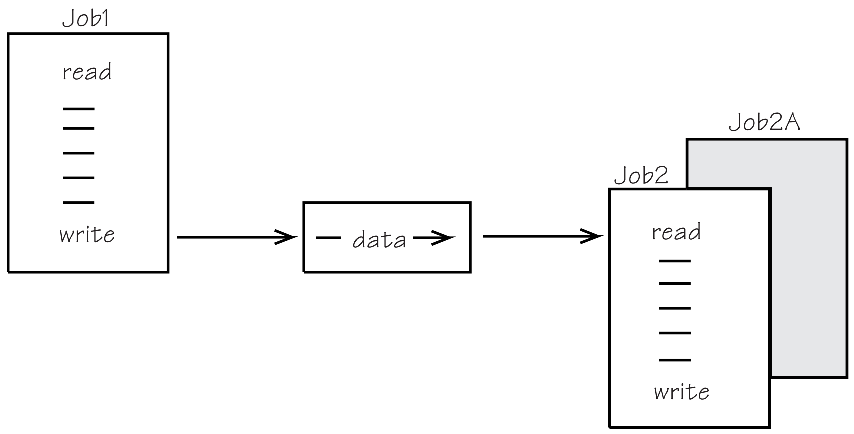

Figure 8. A Writer Job and Two Reader Jobs Using BatchPipes

Running one copy of the writer with the two copies of the reader increases the parallelism that was described in Figure 4 on page 3. Compare Figure 4 with Figure 9, where Job2 and Job2A take turns reading the output from Job1, the faster-running job.

```
<-----Job1------->
 <-----Job2------->
 <-----Job2A------>

____________________________________________
 '  '  '  '  '  '  '  '  '  '  '  '  '  '  '

0     2     4     6     8    10     12    14
```

Figure 9. Timeline Showing Two Reader Jobs Using BatchPipes

The effort required to gain the parallelism represented by Job2 and Job2A is not a large one; the gains in elapsed time can be impressive.

Using a Filter to Make Two Copies of a Data Set

In some cases, you might want two or more readers to read an entire data set written by one job. This situation is different from that in Figure 8, where each of the two readers receives only a portion of the data set. For multiple readers to receive all the records written by one job, use a job that changes how records flow through a pipeline. Such a job is known as a **filter**. For example, in Figure 10 on page 12, Job2 and Job3 both read the entire data set that Job1 writes. The job named Filter writes two copies of each output record; one copy is read by Job2 and one is read by Job3.

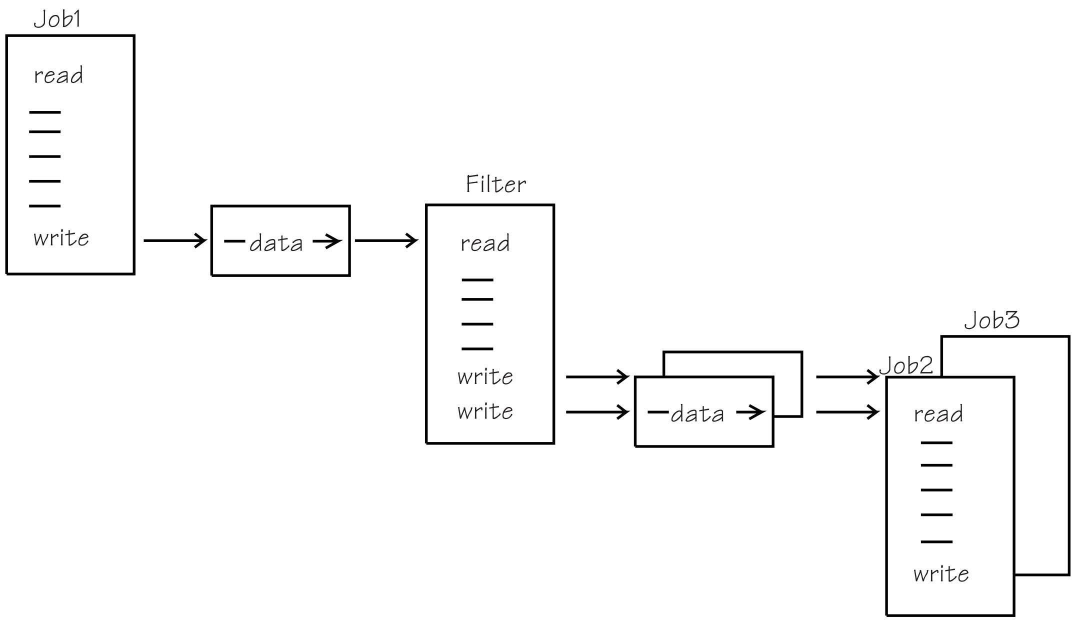

Figure 10. A Pipeline That Includes a Filter that Duplicates Records

Similarly, one of the write requests issued by Filter could be to an external device. In this way, an output data set exists. The process of making a permanent copy of the output records is called **hardening data**.

Keep in mind that all four jobs (Job1, Filter, Job2, and Job3) require all their resources, such as initiators, simultaneously. For help in understanding filters, see *BatchPipes OS/390 Users Guide and Reference*.

Using BatchPipeWorks with BatchPipes Jobs

In a batch environment, there is often a need for additional function in the form of programs that manipulate records or change the contents of records. Your installation might write or buy programs that:

- Write or read selected records
- Reformat records
- Count records
- Write duplicate records to multiple readers
- Harden data

A part of the BatchPipes product, BatchPipeWorks consists of "commands" that you specify on JCL; they can do all of the above and more. They get control "between" an application and a pipe; that is, either after a job writes to a pipe or before a job reads from a pipe. The command, or series of commands, is called a **fitting**; BatchPipes invokes the fitting on behalf of the application.

Using a BatchPipeWorks Fitting to Make Two Copies of a Data Set

There are over 100 BatchPipeWorks commands. To describe how to use them, Figure 11 on page 13 shows an example of how BatchPipeWorks handles the task of writing two copies of a data set to two pipes. Record-by-record, the data flows through Job1 and its fitting, which consists of three BatchPipeWorks commands that:

- Read a record into the fitting
- Write a record to the first pipe
- Write the same record to the second pipe

Job1 and its fitting run in parallel with Job2 and Job3.

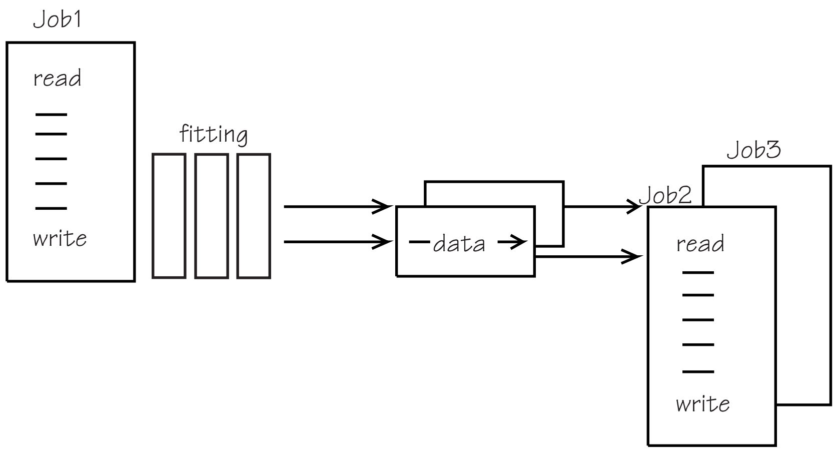

Figure 11. A Pipeline that Includes a Fitting that Duplicates Records

The fitting in Figure 11 is a **writer fitting**. You can also have a **reader fitting**, located between the pipe and a reader, such as Job2. This fitting could change the records as they flow from the pipe to Job2's reader.

Benefits of BatchPipeWorks

The primary benefit of using BatchPipeWorks is that it reduces resources: initiators, virtual and processor storage, and processor cycles, in some cases. Compare the BatchPipeWorks fitting in Figure 11 with the filter in Figure 10 on page 12. Note that the fitting requires one less initiator and associated address space, one less pipe, two fewer connections, and less data movement through pipes.

Additional benefits of BatchPipeWorks are:

- With BatchPipeWorks, it is easy to change what the applications do without having to change the application code. BatchPipeWorks increases your development and prototype capability, as well as your ability to respond to single-instance use of code.
- Because BatchPipeWorks is added to the JCL rather than to the application code, you do not need to recompile, assemble, and linkedit the application code.
- Scheduling of jobs is simplified because fittings are part of a job, rather than being a separate job.
- Fittings that select and exclude records can reduce the number of records that jobs process.

Other Uses for BatchPipeWorks

Through BatchPipes, you can use the BatchPipeWorks commands with jobs that are not in a pipeline. Consider the non-BatchPipes job that needs to process only a subset of the records that it reads. You can have a BatchPipeWorks fitting associated with that job even though it is neither a BatchPipes reader nor a writer. The fitting can select the records, sending to the reader only those records that match the criteria.

Additionally, you can use the BatchPipeWorks commands as a programming language and also in the foreground, under TSO.

For information about how to use BatchPipeWorks, see the *BatchPipes OS/390 BatchPipeWorks Users Guide* and the *BatchPipes OS/390 BatchPipeWorks Reference*.

Customer Application Using BatchPipes

The example in this section is a financial application that runs at a non-IBM installation. The description includes accurate "before" and "after" BatchPipes statistics, as reported from runs in a dedicated environment (that is, there was no other activity on the processor); it does not include information that is considered proprietary. The example is included in this book to help you understand:

- The benefits to an application from using BatchPipes
- The effect using BatchPipes can have on the resources required for an application.

As you read the example, realize that not all applications are appropriate for BatchPipes and gains other applications can realize vary according to the type of processing the jobs do, the size of the data set being handled by BatchPipes, whether the intermediate data set resides on tape or DASD, the logical record length and blocking, the system load, and the amount of parallelism that is possible.

The application in the example consists of two jobs, as shown in Figure 12 on page 15. JobJ reads from four tapes and creates four tapes that become input to JobK. The tapes are demounted and then remounted between JobJ and JobK. Output from JobK goes to four tapes.

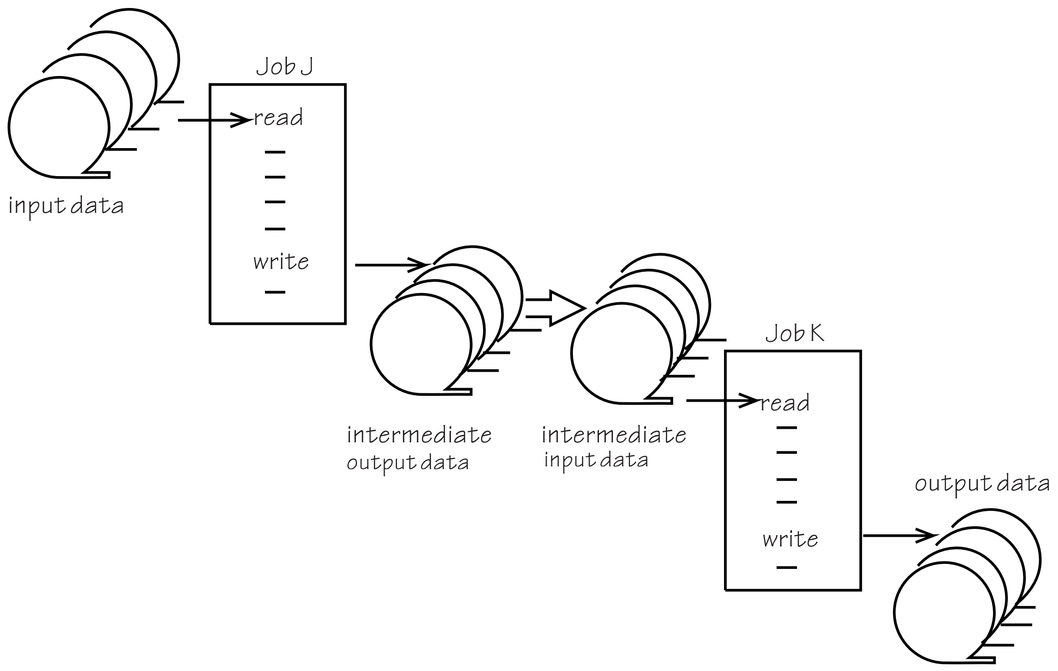

Figure 12. Customer Application before BatchPipes

Important facts about the application are:

- Total elapsed time is 1896 seconds
- Total processor time used is 121.9 seconds (reported by RMF as "busy time.")
- Total tape mounts required is 16
- Total EXCP count is 216K.

The JCL for JobJ and JobK included the following DD statement:

```
//DD1    DD DSNAME=DW.ACCT.DATA,DISP=SHR,UNIT=TAPE,VOL=SER=TAPE01,
//       LRECL=1354,BLKSIZE=32760,RECFM=VB,...
```

By adding the SUBSYS parameter to the JCL, an application developer enabled the two jobs to use the BatchPipes subsystem named *BP02*.

```
//DD1    DD DSNAME=DW.ACCT.DATA,DISP=SHR,UNIT=TAPE,VOL=SER=TAPE01,
//       SUBSYS=BP02,
//       LRECL=1354,BLKSIZE=32760,RECFM=VB,...
```

The system ignores the UNIT, VOL, and DISP parameters. The subsystem named BP02 now intercepts the write requests from JobJ and the read requests from JobK; the two jobs run concurrently, as Figure 13 shows.

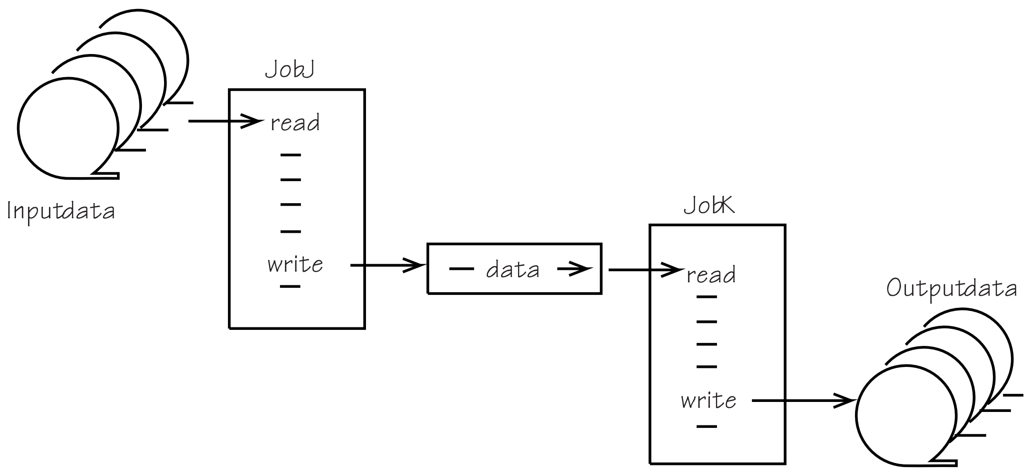

Figure 13. Customer Application with BatchPipes

Figure 14 summarizes the important “before” and “after” BatchPipes facts for the two jobs and shows the improvements that came with BatchPipes:

Figure 14. Comparing Key Statistics Before and After BatchPipes

Item	**Before**	**After**	**Change**
Total elapsed time	1896	714	-62%
Total processor time used	121.9	120.3	-1.3%
Tape mounts	16	8	-50%
Total EXCP count	216K	109K	-50%

Another change, not visible in the table, is how the system divides the “total processor time used” between “system” time and “application” time. With BatchPipes, the system assigns more of the time used to write and read the data to the application; therefore, you can expect the application time to increase. In the example here, the reported amount of total processor time assigned to the application increased **by 27%**, even though the total processor time used remained approximately the same.

This customer example shows the gains realized for an actual application after the minimal JCL change. We hope it will encourage you to look hard at what BatchPipes can do for applications in your batch environment.

Appendix: Overview of BatchPipes Functions

Operating System

- MVS/ESA Version 5.0 or later

Access Methods

- Support for sequential QSAM, sequential BSAM, and GSAM-BSAM (although BatchPipes does not support requests for checkpoint and restart.)
- ASA and machine print control character formats are supported.
- RECFM values F, FA, FM, FB, FBA, FBM, V, VA, VM, VB, VBA and VBM

Security

- RACF (or an equivalent product) check for use of BatchPipes subsystem defined through FACILITY class
- RACF authority to issue BatchPipes commands
- RACF authority to access data set
- RACF authority to control the use of fittings.

BatchPipes Capability

- Types of writer/reader connections
  - One writer to one reader
  - Many copies of writer to one reader
  - One writer to many copies of reader
  - Many copies of writer to many copies of reader
- Up to 125 pipes associated with one BatchPipes subsystem
- Up to 250 pipe connections per subsystem for writers and readers that have allocated the pipe data sets
- Multiple unique BatchPipes subsystems on the same MVS image
- Scheduling options are:
  - Open synchronization; that is, BatchPipes does not build pipe connections until specified number of writers and readers have opened the pipe data set.
  - Close synchronization; that is, BatchPipes does not close pipe connections until all users of the pipe have closed the pipe data set.
  - Open-now option; that is, BatchPipes builds a pipe connection for a job without waiting for a partner to open the pipe data set.

Commands

- The BatchPipes STATUS command that displays information about jobs using pipes, such as:
  - Number of readers and writers using pipes
  - Job-pipe-job relationships in a jobstream
- BatchPipes commands: CANCEL, STOP, DISPLAY, SET, DUMP, TRACE, HELP, and EOF.

Documentation

Figure 15. Basic Material: Unlicensed Publications

Publication Title	**Form Number**
IBM BatchPipes OS/390 V2R1 Introduction	GA22-7459
IBM BatchPipes OS/390 V2R1 Users Guide and Reference	SA22-7458
IBM BatchPipes OS/390 V2R1 BatchPipesWorks Reference	SA22-7456
IBM BatchPipes OS/390 V2R1 BatchPipesWorks User Guide	SA22-7457

Monitoring

- Threshold alerts
- BatchPipes SMF records
- ISPF BatchPipes monitor
- BatchPipes query service interface

Communicating Your Comments to IBM

IBM BatchPipes OS/390
Introduction

Publication No. GA22-7459-00

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

If you are mailing a reader's comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF at the back of this book.
- If you prefer to send comments by FAX, use this number:
  - FAX: (International Access Code)+1+845+432-9405
- If you prefer to send comments electronically, use one of these network IDs:
  - IBM Mail Exchange: USIB6TC9 at IBMMAIL
  - Internet e-mail: mhvrcfs@us.ibm.com
  - World Wide Web: http://www.ibm.com/s390/os390/

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies

Optionally, if you include your telephone number, we will be able to respond to your comments by phone.

Reader's Comments — We'd Like to Hear from You

IBM BatchPipes OS/390
Introduction

Publication No. GA22-7459-00

You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Today's date: ____________________

What is your occupation?

Newsletter number of latest Technical Newsletter (if any) concerning this publication:

How did you use this publication?

[ ]	As an introduction	[ ]	As a text (student)
[ ]	As a reference manual	[ ]	As a text (instructor)
[ ]	For another purpose (explain)		

Is there anything you especially like or dislike about the organization, presentation, or writing in this manual? Helpful comments include general usefulness of the book; possible additions, deletions, and clarifications; specific errors and omissions.

Page Number: Comment:

Name	Address
Company or Organization	
Phone No.	

Reader's Comments — We'd Like to Hear from You
GA22-7459-00

Cut or Fold
Along Line

IBM®

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY 12601-5400

Fold and Tape **Please do not staple** Fold and Tape

GA22-7459-00

Cut or Fold
Along Line

File Number: S370/S390-34
Program Number: 5655-D45

GA22-7459-00