WebSphere Business Integration Message Broker for z/OS V5

VSAM Nodes

Performance report

Version 1.0

December 2004

Dave Fisher

IBM UK Laboratories
Hursley Park
Winchester
Hampshire
SO21 2JN

Take Note!

Before using this report please read the general information under "Notices".

First Edition, December 2004

This edition applies to Version 1.0 of *WBI Message Broker for z/OS V5.0 VSAM Nodes.* The Performance Report is based on a Beta version of the code for SupportPac IA13.

Notices

This report is intended for Architects, Systems Programmers, Analysts and Programmers wanting to verify the installation, and understand the performance characteristics, of **WBI Message Broker for z/OS V5.0 VSAM Nodes**. The information is not intended as the specification of any programming interfaces that are provided by WebSphere MQ or WBI Message Broker for z/OS – V5. It is assumed that the reader is familiar with the concepts and operation of WebSphere Business Integration Message Broker V5.

References in this report to IBM products or programs do not imply that IBM intends to make these available in all countries in which IBM operates.

Information contained in this report has not been submitted to any formal IBM test and is distributed "as is". The use of this information and the implementation of any of the techniques is the responsibility of the customer. Much depends on the ability of the customer to evaluate these data and project the results to their operational environment.

The performance data contained in this report was measured in a controlled environment and results obtained in other environments may vary significantly.

Trademarks and service marks

The following terms, used in this publication, are trademarks or registered trademarks of the IBM Corporation in the United States or other countries or both:

Enterprise Storage Server

IBM

SupportPac

WebSphere

WebSphere Business Integration Message Broker

WebSphere MQ

WebSphere MQ Integrator

z/OS

zSeries

Other company, product, and service names may be trademarks or service marks of others.

Summary of Amendments

Date	Changes
17 December 2004	Initial version.

Table of contents

1 Introduction

VSAM Nodes

VSAM Nodes are a set of WebSphere Business Integration Message Broker (WBIMB) Message Processing nodes that you can use within a message flow to make use of VSAM files. They are implemented on the z/OS platform broker only.
VSAM Nodes and supporting documentation is available for download as Product Extension SupportPac IA13 from
http://www.ibm.com/software/integration/support/supportpacs/category.html#cat3

This document

SupportPac IA13 includes sample flows with descriptions. This document is a Performance Report which presents and discusses the performance of the VSAM Nodes using those supplied sample flows. It is based on measurements of a Beta release carried out at the IBM Hursley Performance lab.

1.1 Results Overview

The results show that IO rates are the determining factor for throughput and that keeping a file open has a significant impact on CPU cost. The interface to VSAM is such that there is little user control over buffering and thus the underlying VSAM read and write performance. In particular BUFND and BUFNI are not available. The IDCAMS DEFINE and ALTER parameter BUFFERSPACE is the only user controllable parameter.

Also, as is usual for workloads using WMQ persistent messages, placement of the WMQ active logs on best performing available DASD will maximize throughput.

CPU costs using WMQ persistent messages are not much different to those using nonpersistent messages.

1.2 Workloads

All the results presented in this document are from two basic workload types, those driven by:

1. **A single application** which MQPUTs a WMQ message to the MQInput node of a sample flow and then MQGETs a WMQ message from an MQOutput node of that sample flow in a loop. In this case although additional instances are defined for the flows deployed for these evaluations they are not being exploited.

2. **5 such applications driving a sample flow with 3 additional instances.** This shows what scalability might be possible when exploiting additional instances as there are always more potential message flows than can be executed concurrently.

Workloads all use:

- Flows configured with 3 additional instances.
- WMQ input and output messages which are either all nonpersistent or all persistent.
- VSAM KSDS's all starting with 65145 records (unless otherwise stated) of the type (32 bytes with 8 byte keys) used in the samples. This results in BUFFERSPACE of 37376.
- Enterprise Storage System (ESS)-F20 DASD configured as 3390-9's for both the VSAM KSDS's and WMQ logs. Improved throughput would be expected from later DASD models.
- 4-engine LPAR of a zSeries 2084-332 approximately equivalent to a 2084-304. No significant other work was done on this system during measurements.

We present the results in detail in the following pages and a fuller workload description in Measurement environment and methodology.

The application, OEMPUTX, is available in SupportPac IP13 at
http://www.ibm.com/software/integration/support/supportpacs/category.html#cat2

1.3 Message rate, CPU cost

The Message rates and CPU costs in this document are obtained from output from the driving OEMPUTX batch application(s) and defined as follows:

Message rate: The number of message flows executed per second.

Driver program(s) put a request WMQ message to the message flow input queue, the flow executes and terminates by sending a WMQ message to an output queue. The driver program gets the first available WMQ message from the output queue. Message rates are expressed in flow messages per second but use request/reply *pairs* of WMQ messages.

CPU Cost, expressed as CPUmicroseconds (2084-304) per message for these address spaces:

- Queue Manager
- Message Broker
- Driving Application(s)

These CPU costs can be translated from the measured 2084-304 to another zSeries system by using the Large Systems Performance Reference (LSPR) tables from http://www.ibm.com/servers/eserver/zseries/lspr/zSerieszOS.html.

This example shows how to estimate the CPU cost for a zSeries 2064-1C5:

1. The LSPR gives the **2064-1C5** an Internal Throughput Ratio (ITR) of **1.47** (this is for a "Mixed Workload", which we found best fits Message Broker in our environment).
2. As the 1C5 is a 5-way processor, the single engine ITR is

   2.35 / 5 = **0.47**
3. The "Mixed Workload" ITR of the **2084-304** used for these measurements is **3.60**.
4. The 304 is a 4-way processor. Its single engine ITR is

   3.60 / 4 = **0.90**
5. The **2064-1C5 / 2084-304 single engine ratio** is

   0.47 / 0.90 = 0.52 approx

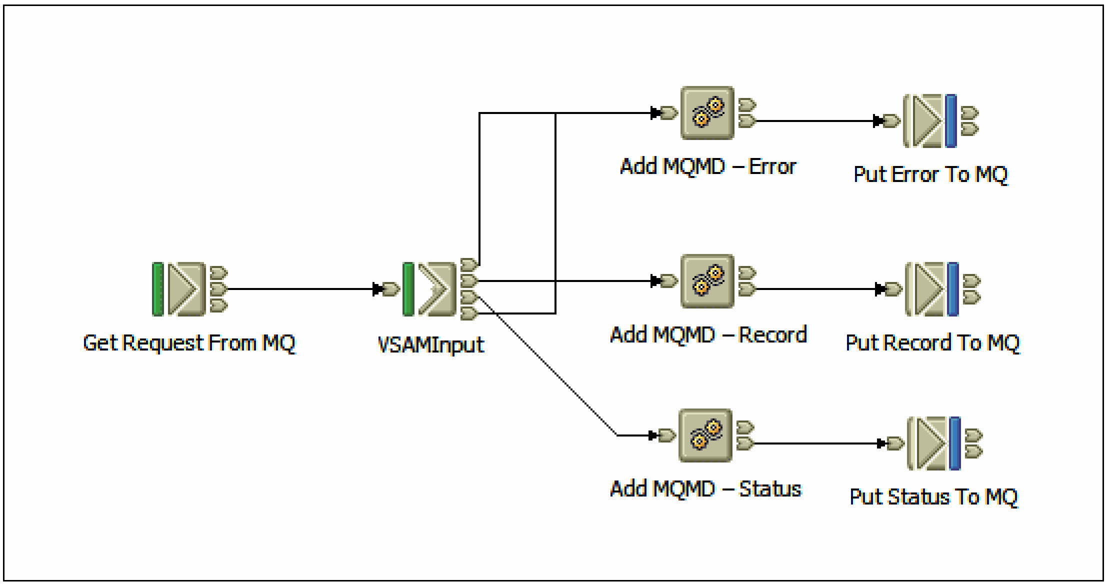
   this means that a single engine of a 2084-304 is nearly twice as powerful as that of a 2064-1C5.
6. Take a CPU cost of interest from this report, say **x** CPUmicrosecs(2084-304) per message, then the equivalent on a 2064-1C5 will be

   x / 0.52 µsec/msg

2 Measurement Results

2.1 VSAMInput Node – Sample 1

Sample1 message flow (see the separately supplied Sample documentation for full details)



The driving application generates a single input message are of the form

<VSAM><Request><FileName>//'DATA.IA13.VSAM.KSDS1'</FileName></Request></VSAM>

The Sample1 flow used in this way will

a) Put no errors to the 'Put Error To MQ' queue.

b) Put one message for each record in the KSDS file to the 'Put Record to MQ' queue, which are NOT consumed in this evaluation.

c) Put a single message to the 'Put Status To MQ' queue at flow execution end.

Note that the 'Get Request From MQ' queue property 'Order Mode' is set to 'By Queue Order'. This means that messages can only be passed to the VSAMInput mode one at a time whatever the additional instances setting may be.

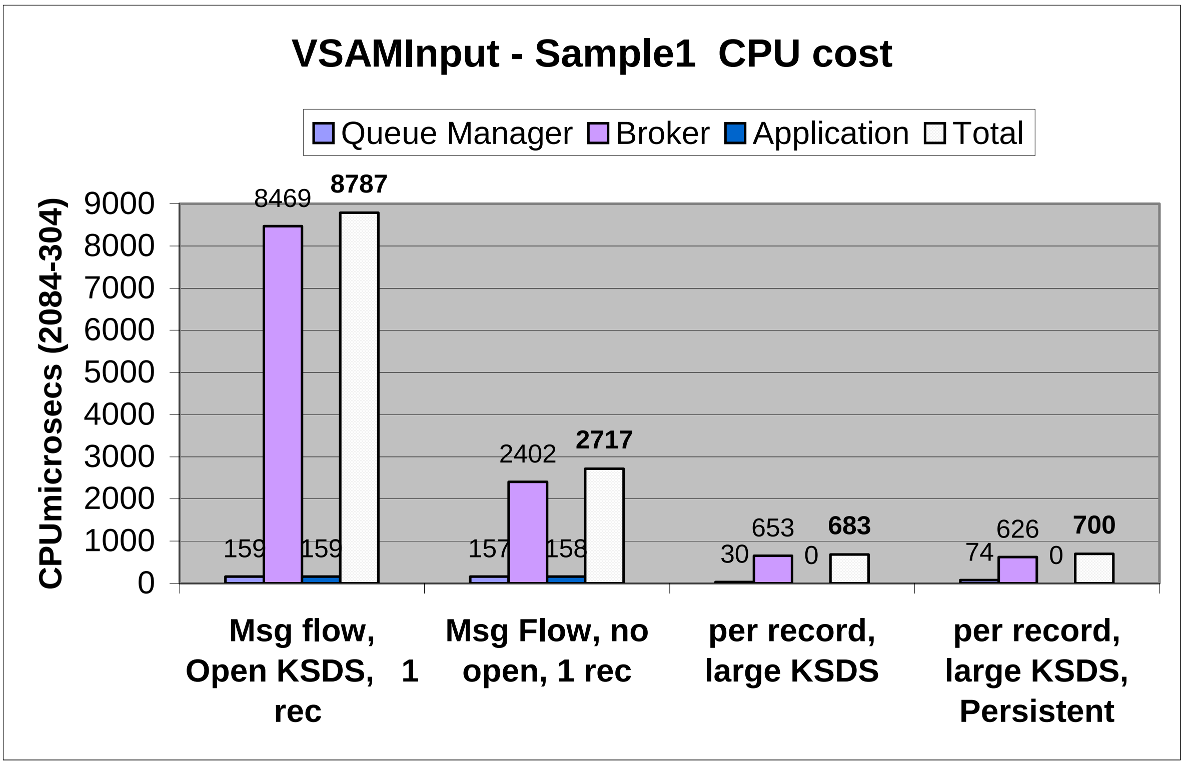
The VSAMInput node reads only a single file at a time. However, VSAMInput is a broker input node. This means that additional instances are exploited for flow paths continuing out from a VSAMInput node.

There were 65145 records in the 'large KSDS', with this many records the cost to open the file is not significant.

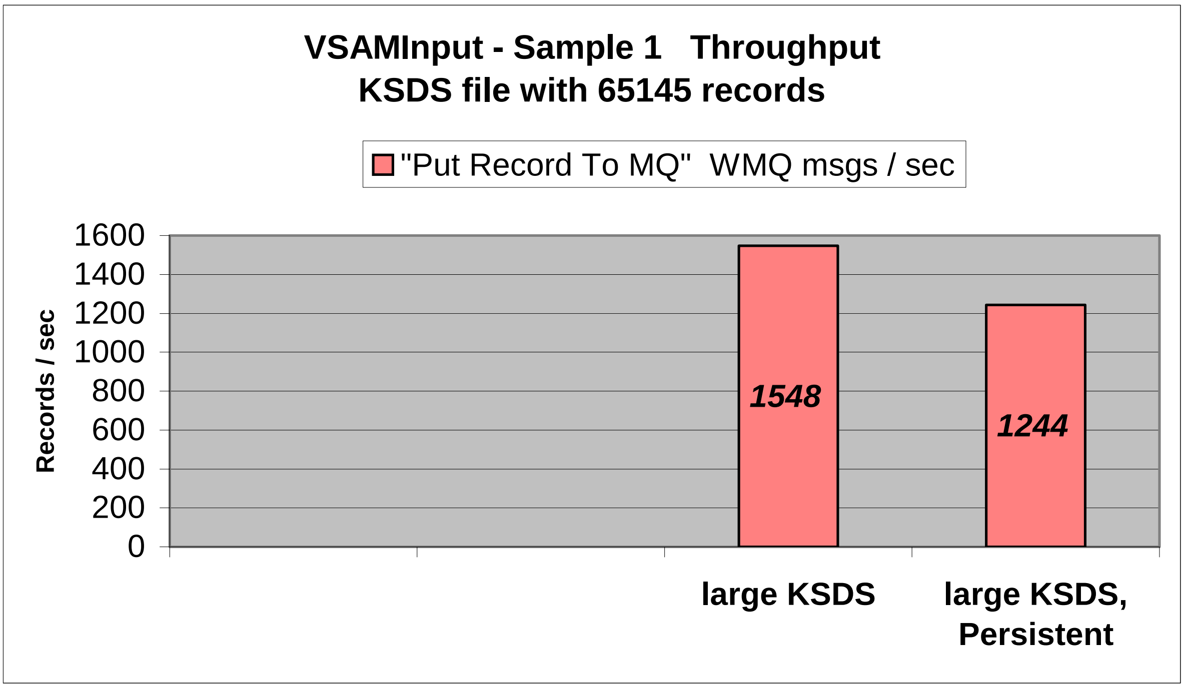
Costs to open and process a KSDS with just one record were also obtained.

.

2.1.1 Results

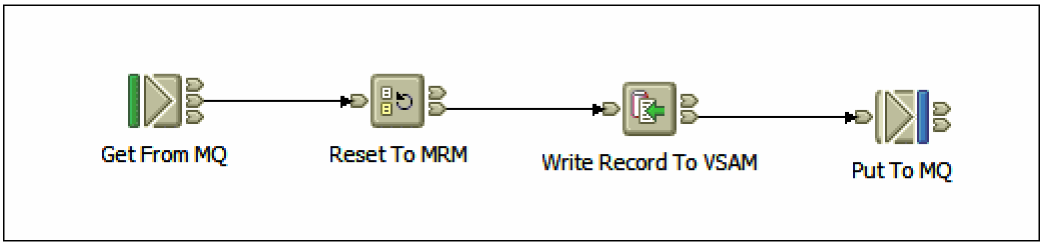


Throughput will clearly depend on the size and structure of the particular VSAM file and the IO performance of the overall zOS system with its particular DASD subsystem. Throughput on another system may differ significantly.



2.2 VSAMWrite Node – Sample 2B

Sample 2B message flow (see the separately supplied Sample documentation for full details)



The messages input to this flow were of the form:

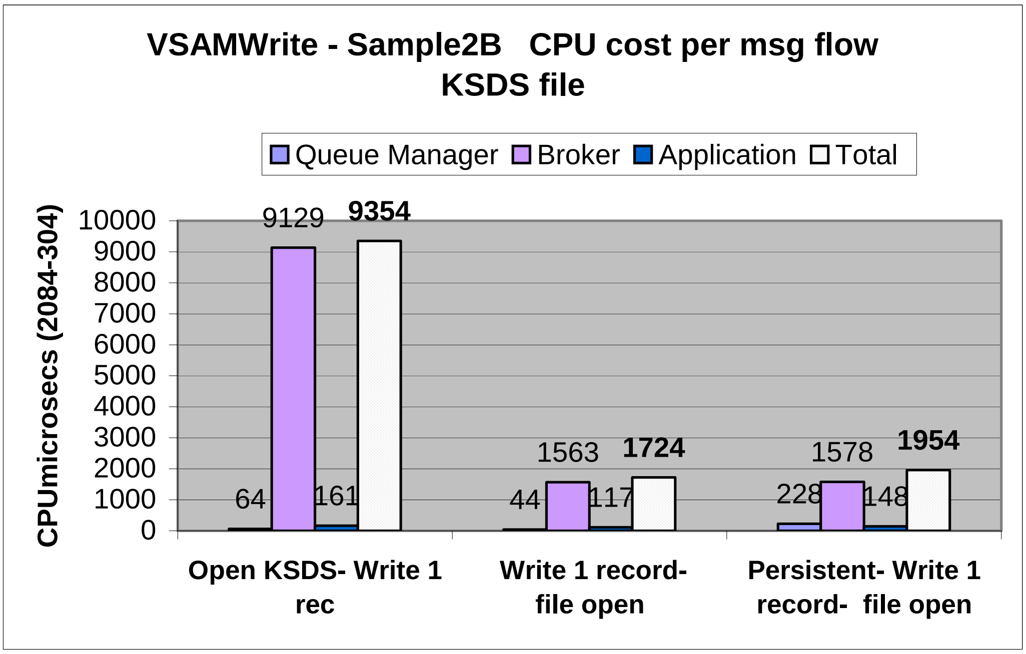
```
<msg_SampleStruct><intField1>000</intField1><intField2>-4</intField2><charField1>KEY_0001</ch
arField1><charField2>THE ????? RECORD</charField2></msg_SampleStruct>
```

The negative intField2 value allows such a record to be deleted by the Sample 5 flow containing a VSAMDelete node.

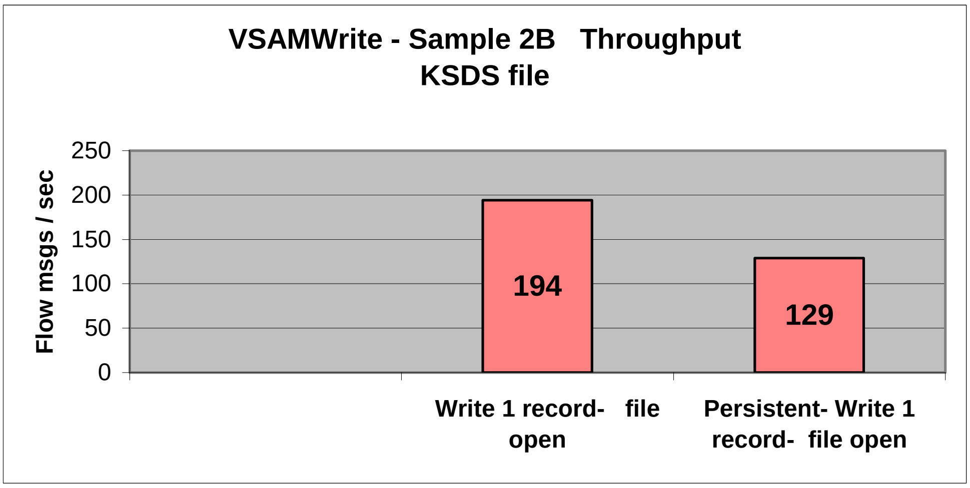
2.2.1 Results

All VSAM Nodes hold the file open for an interval if "Retain File" is set to "Until Idle" in the Properties Advanced section. Sample 2B does use this setting. The first message flow after an interval must open the required VSAM file and uses much more CPU.

The costs per record written were the average of 27 records written to the already open KSDS file of about 65000 records. The keys were spread across the KSDS.

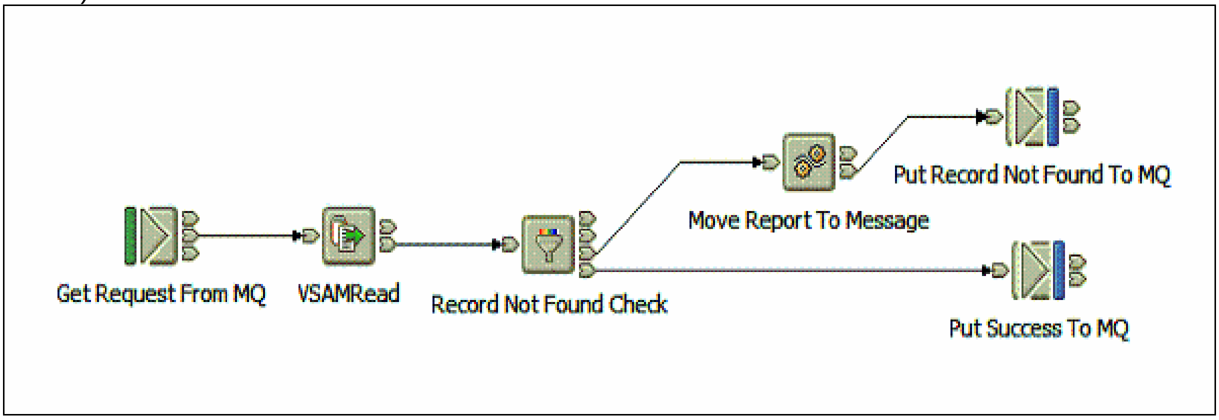


The achieved throughput was as follows. Results from other flows indicate that persistent message driven flow throughput could be closer to that for nonperistent where additional instances are exploited by having multiple driving applications.


VSAMWrite - Sample 2B Throughput
KSDS file
Flow msgs / sec
250
200
150
100
50
0
194
129
Write 1 record- file open
Persistent- Write 1 record- file open

2.3 VSAMRead Node – Sample 3

Sample 3 message flow (see the separately supplied Sample documentation for full details)



Input messages were of the form

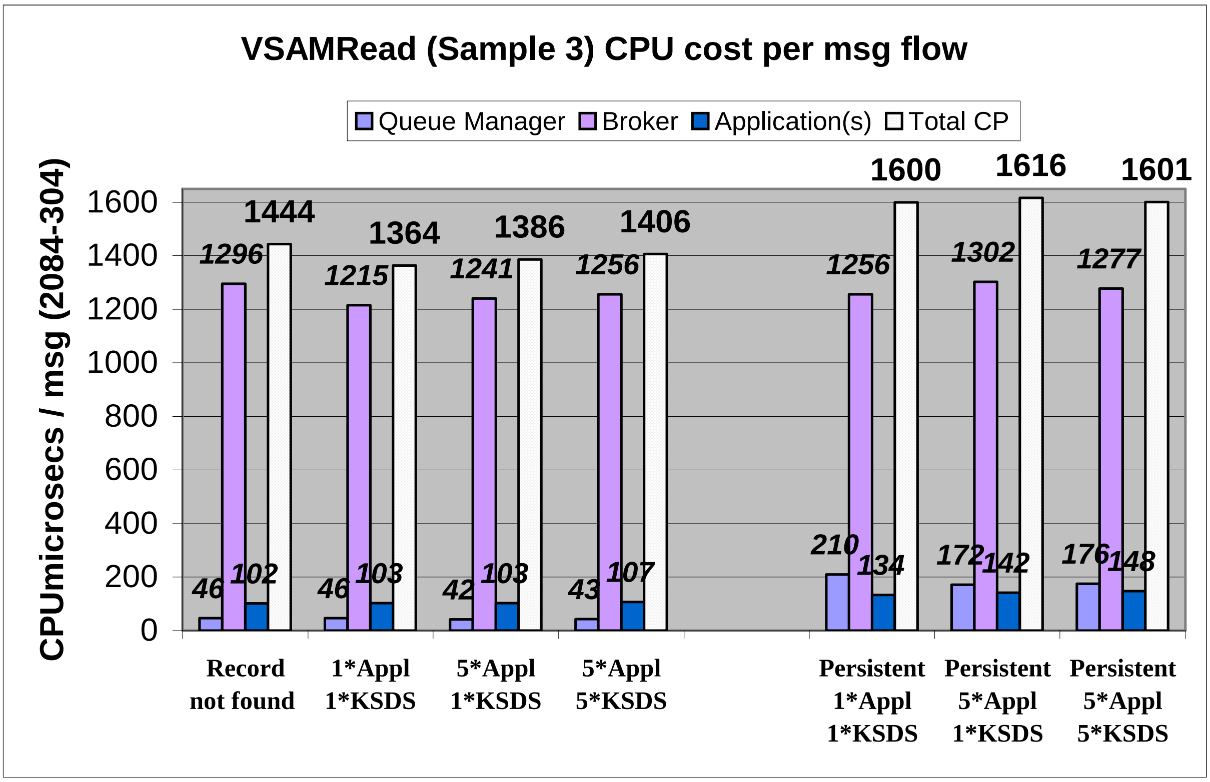
```
<VSAM><Request><FileName>//'DATA.IA13.VSAM.KSDS1'</FileName><Position><Key>
RECORD_1</Key></Position></Request></VSAM>
```

There were 29 messages each with a unique key value. The keys were spread across the KSDS. All records were found unless otherwise stated.

2.3.1 Results

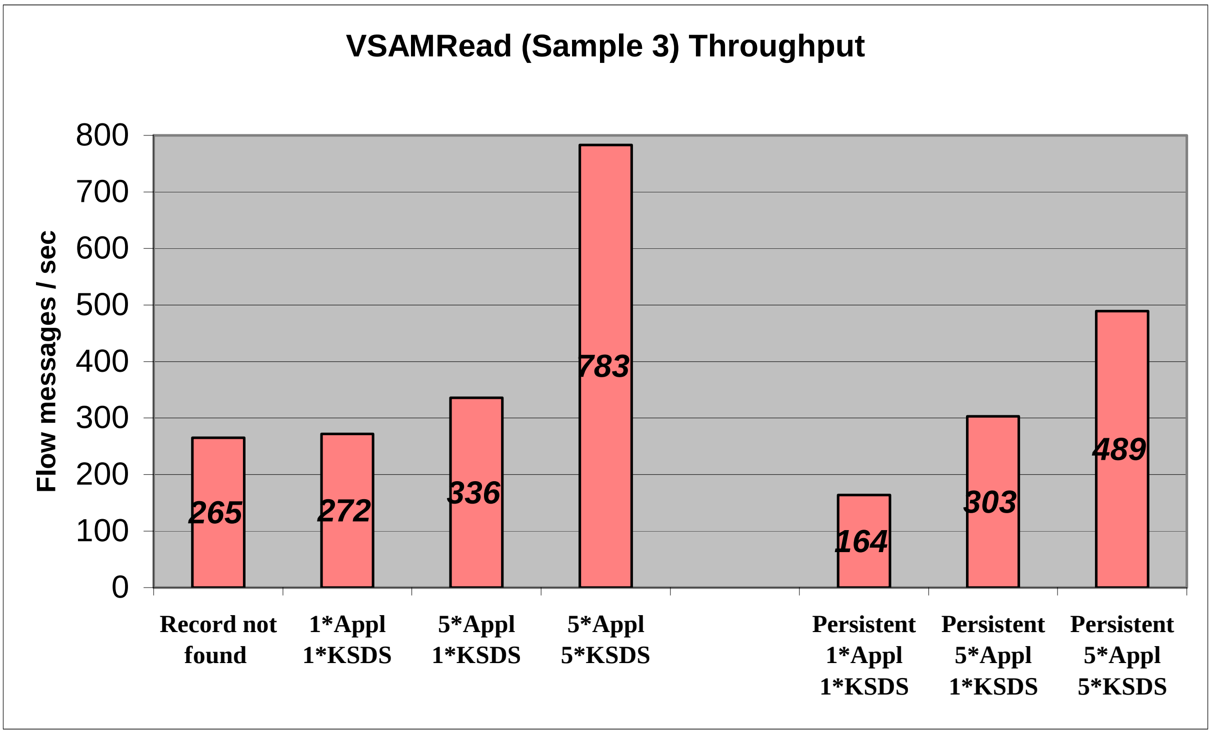
The Broker CPU cost to process one message flow which opens a KSDS and reads one record was 7750 CPUmicrosecs (2084-304).

This first time cost is not significant in the following measurements which are the average cost for 10000 message flows as the Sample 3 VSAMRead node holds files open.



The results indicate that CPU costs are not significantly changed by exploitation of multiple instances nor multiple files (of the same type and size).

Throughput is limited by IO not by CPU. Multiple concurrent use of the same KSDS file can increase throughput, but quickly becomes totally IO bound. Multiple concurrent use of a set of KSDS files spreads the IO load and allows reasonable throughput scalability.



2.3.2 Throughput varying BUFFERSPACE

The KSDS's used to produce all other results in this document were defined such that BUFFERSPACE was defaulted (to 37376 given the rest of the definition). Here we show some effects from using the IDCAMS ALTER command to change BUFFERSPACE as there appears to be no other file tuning option available.

Altering BUFFERSPACE could have an impact on the performance of other programs using that file. However, it is likely that many if not all other such programs will have bufferpool or DD card tuning options like BUFND and BUFNI available. In general use of such options overrides the value of BUFFERSPACE. See "Access Method Services for the Integrated Catalog Facility" SC26-4906 for details of the BUFFERSPACE attribute.

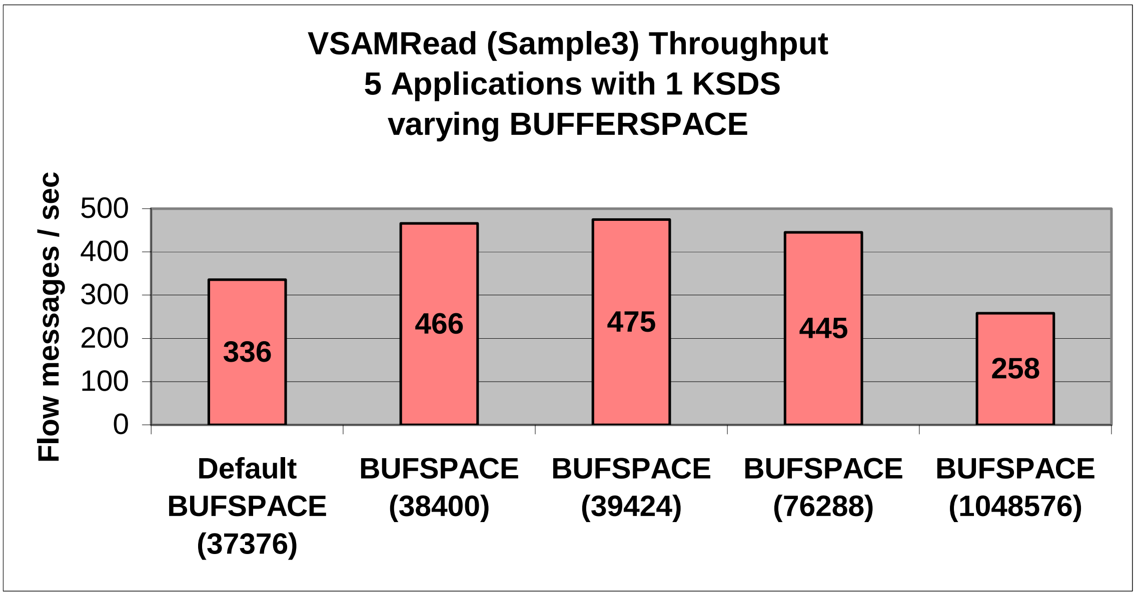
An IDCAMS LISTCAT for the KSDS shows

- For the DATA component attributes
    - BUFSPACE-----------37376 CISIZE-------------18432
- And for the INDEX component attributes
    - BUFSPACE---------------0 CISIZE---------------512

The default BUFSPACE of 37376 is that required for 2 DATA and 1 INDEX CI's.

The chosen values for the ALTER BUFFERSPACE command, which can only change the DATA component attribute, were:

- 38400
  - Because this adds 1024 bytes which is space for exactly 2 more INDEX CI's than the default (and not big enough to allow more DATA CI's)
- 39424
  - Because this is adds a further 1024 bytes which is exactly another 2 more INDEX CI's (and still not big enough to allow more DATA CI's)
- 76288
  - Because this adds a further 36864 bytes which is space for exactly 2 more DATA CI's
- 1048576 (1MB)
  - Because that is a big round number that might be expected to give as good if not better performance than smaller values.

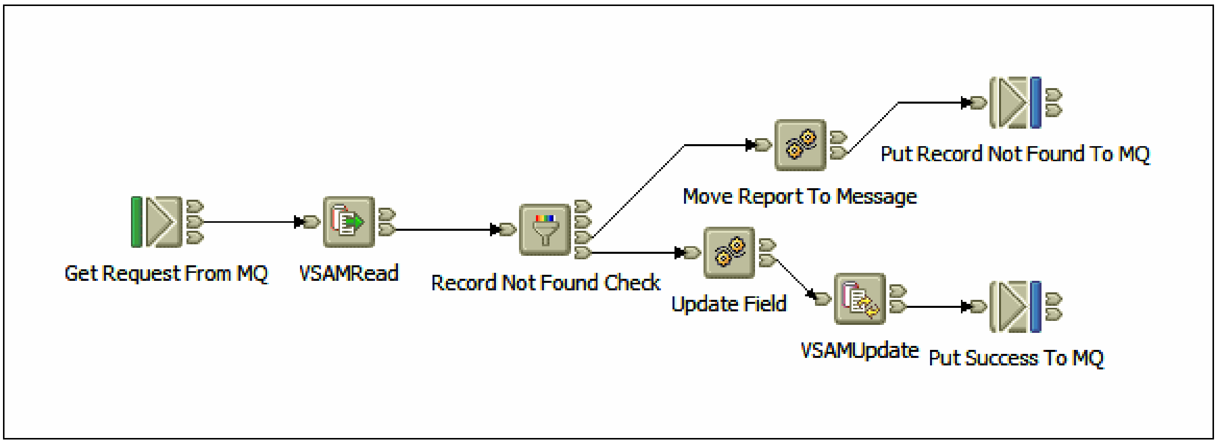


The above results indicate that having more INDEX CI's is the most important tuning consideration, at least for VSAMRead with many applications using a single KSDS file.

CPU costs were not greatly changed for any of these experiments.

2.4 VSAMUpdate – Sample 4

Sample 4 message flow (see the separately supplied Sample documentation for full details)

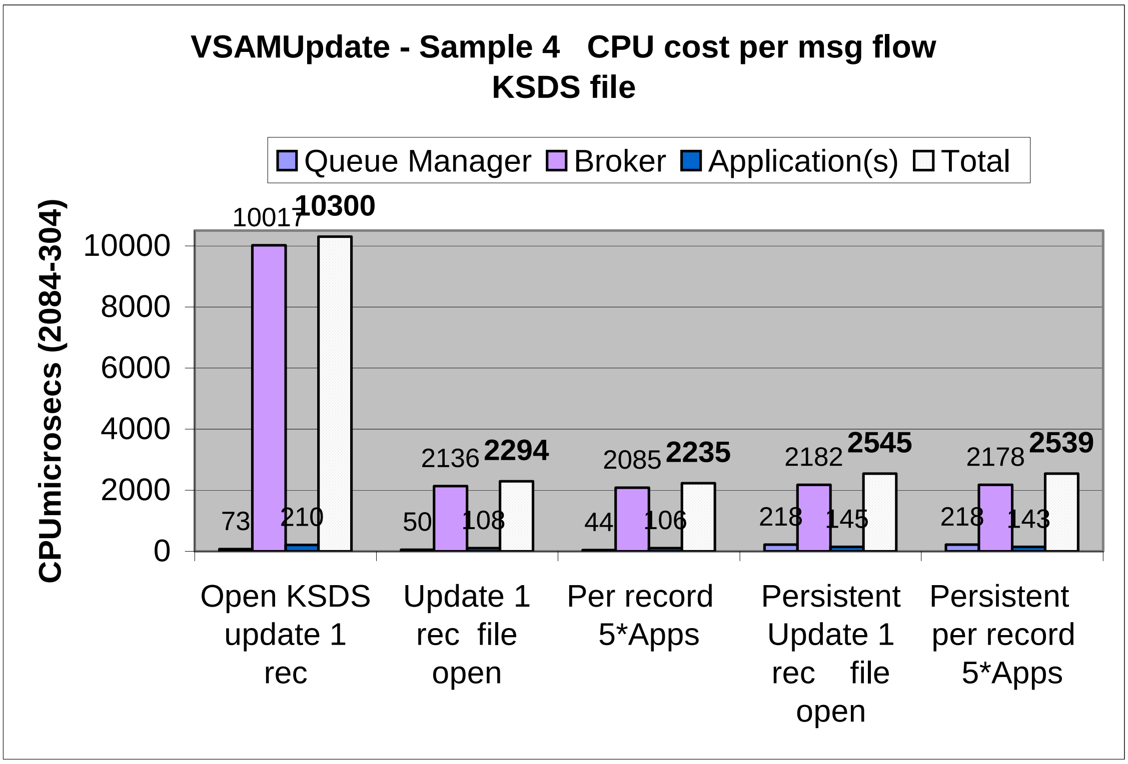


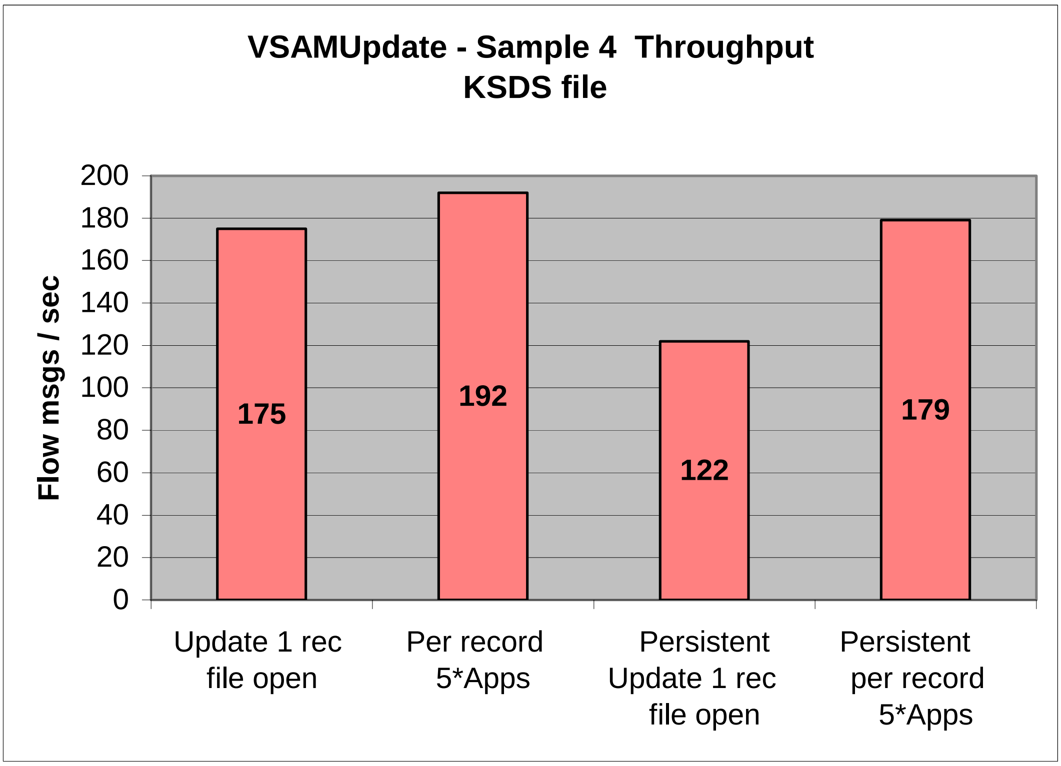
Input messages were of the form

<VSAM><Request><FileName>//'DATA.IA13.VSAM.KSDS1'</FileName><Position><Key>RRR_r019 </Key></Position></Request></VSAM>

There were 27 messages each with a unique key value. The keys were spread across the KSDS.

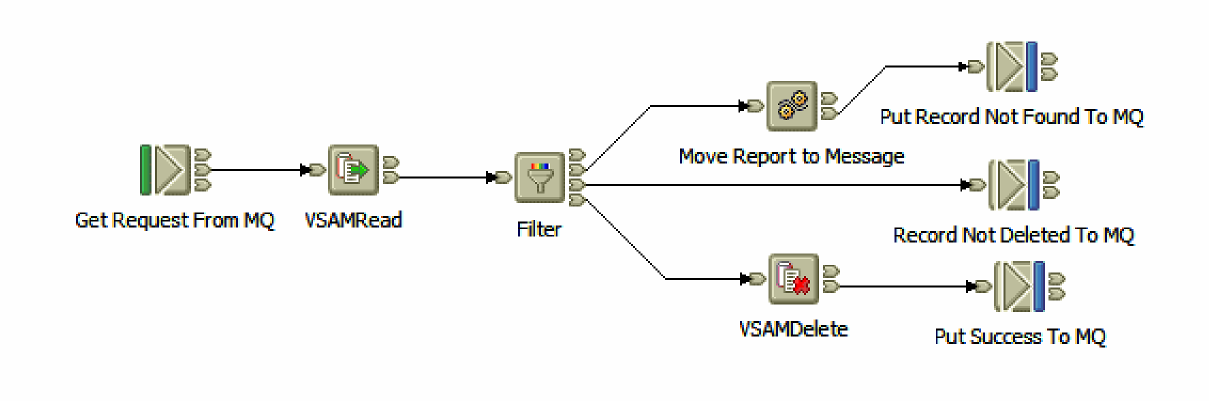
2.4.1 Results




VSAMUpdate - Sample 4 Throughput
KSDS file
Flow msgs / sec
200
180
160
140
120
100
80
60
40
20
0
175
192
122
179
Update 1 rec file open
Per record 5*Apps
Persistent Update 1 rec file open
Persistent per record 5*Apps

2.5 VSAMDelete – Sample 5

Sample 5 message flow (see the separately supplied Sample documentation for full details)

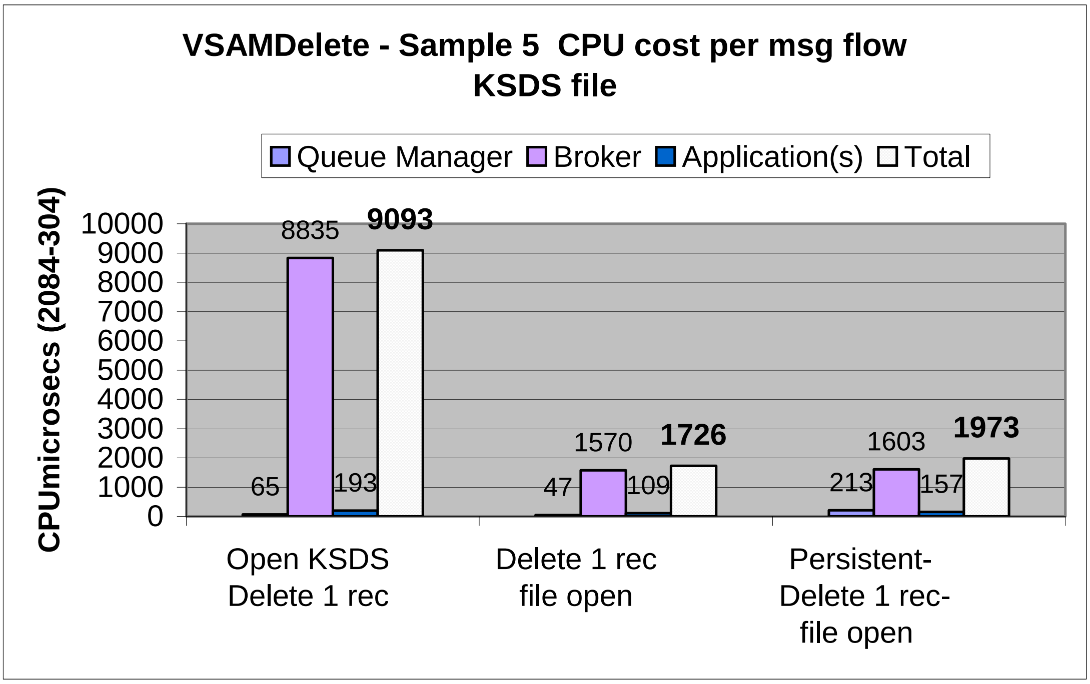


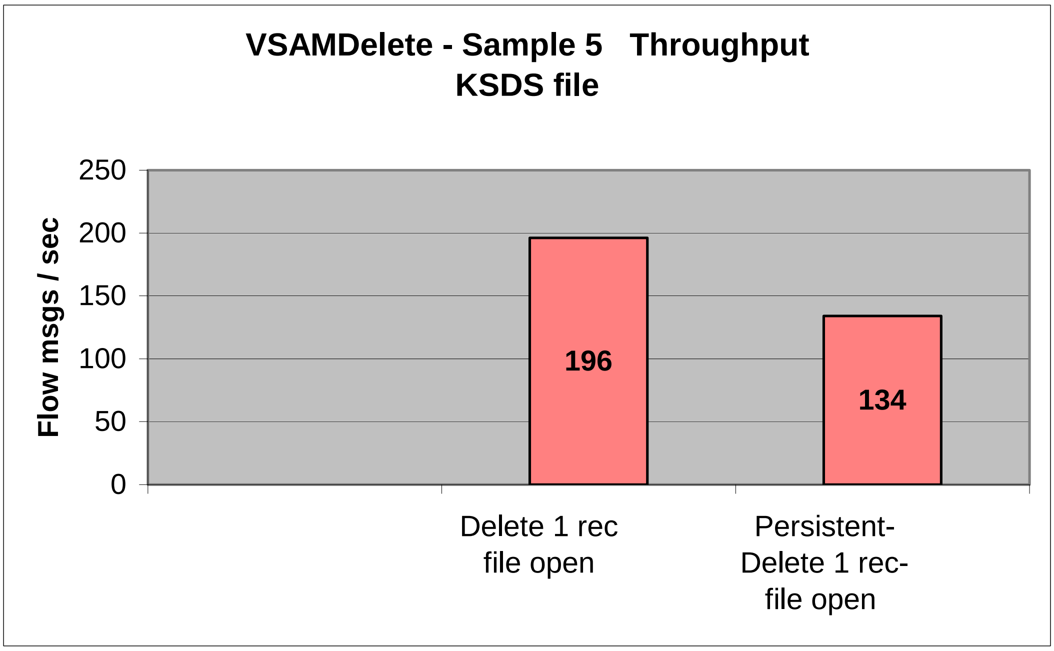
Input messages were of the form

<VSAM><Request><Position><Mode>KEY_EQ</Mode><Key>RECORD_1</Key><KeyType>Parser String</KeyType></Position></Request></VSAM>

The costs per record deleted were the average of 27 records deleted from the already open KSDS file of about 65000 records. The keys were spread across the KSDS.

2.5.1 Results




VSAMDelete - Sample 5 Throughput
KSDS file
Flow msgs / sec
250
200
150
100
50
0
196
134
Delete 1 rec file open
Persistent-Delete 1 rec-file open

3 Measurement environment and methodology

All the measurements consist of a simple message flow with one or more of the same batch application driving a SupportPac IA13 supplied sample VSAM Nodes flow.

3.1 Application used to drive the workloads

Each sample flow is driven by applications each of which

1. loop round
    a. MQPUT a WMQ ‘request’ message on the input queue at ‘Get Request From MQ’
    b. MQGET the first available WMQ ‘reply’ message from the relevant MQOutput node queue. The samples are such that this ‘reply’ message has the same persistence as the ‘request’ message.
2. Use
    a. Out-of-syncpoint calls for non persistent messages.
    b. MQCMIT after each MQPUT and MQGET for persistent messages.
    c. 200 byte input messages. An example of the message contents is shown in each sample result section.
    d. Output message length is defined by the flow.

3.2 Hardware and Software

The hardware configuration was:

- **CPU:** 4-CPU logical partition (LPAR) of a zSeries 990 (2084-332). CPUs were defined as floating but there were always 4 physical CPUs available. Its capacity is similar to that of a 2084-304.
- **DASD**:VSAM files on FICON-connected Enterprise Storage Server (ESS) Model F20.
  WebSphere MQ logs on FICON-connected ESS Model F20.

Software levels were:

- z/OS 1.6.
- WebSphere MQ 5.3.1.
- WebSphere Business Integration Message Broker 5.0.1. CSD3

End of Document