

***OSAM -  
“the Healthy  
Alternative to VSAM”***

*Alan Cooper  
EMEA Technical Sales  
Manchester, UK*



# Origins of OSAM

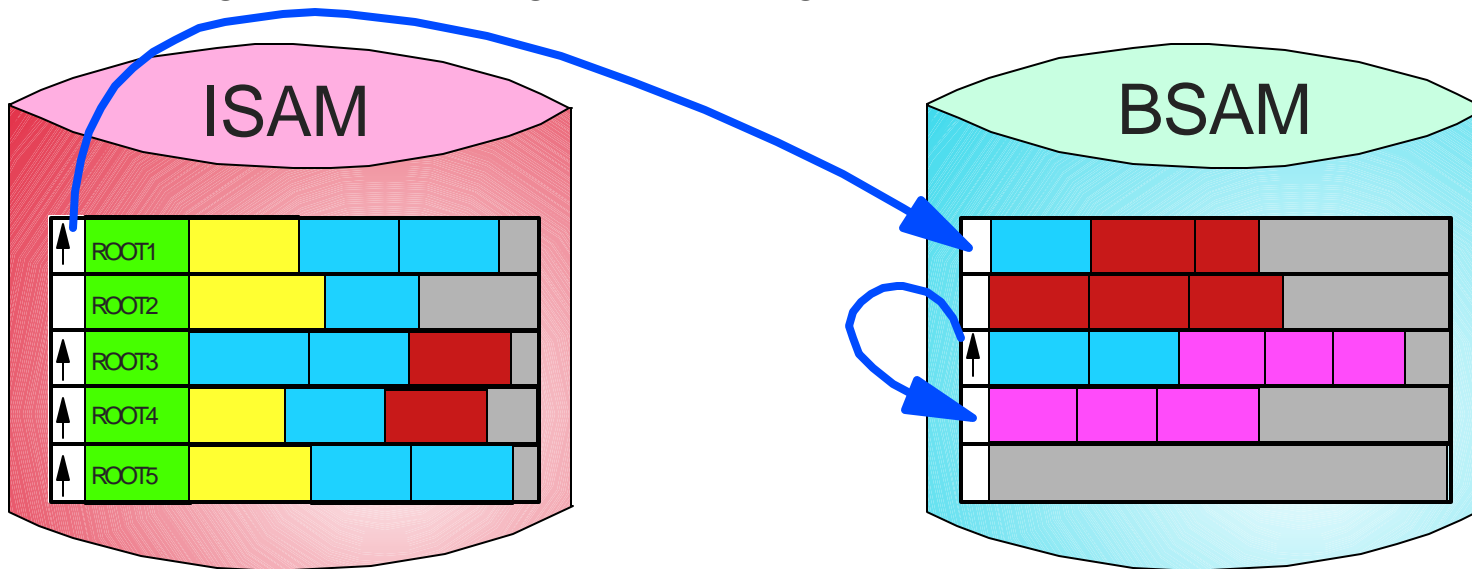


## Before VSAM there was ISAM

- ▶ Indexed Sequential Access Method

## HISAM DB

- Root at start of ISAM logical record
- Root Key = ISAM record key
- Dependent segments stored, in sequence, following root
  - ▶ as many as will fit in ISAM logical record
- Remaining dependent segments must go into a BSAM overflow dataset



▶ The IMS code written to access segments in the BSAM overflow

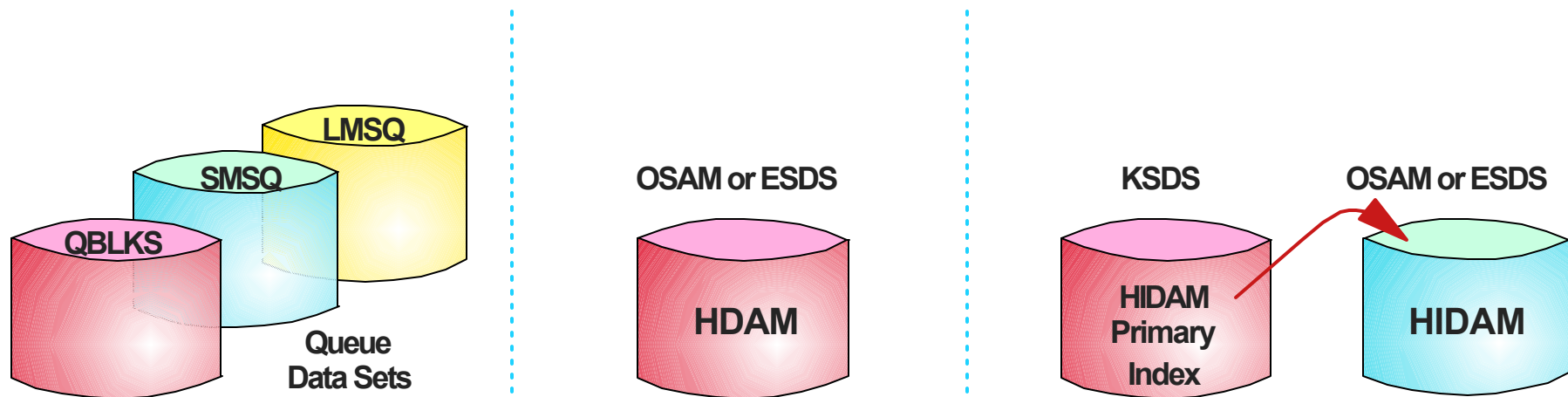
➔ **OSAM (Overflow Sequential Access Method)**

# Where IMS Now Uses OSAM



## Message Queue Data Sets

## Optionally for HDAM and HIDAM data component



- Some customers say “VSAM is an IBM strategic access method, but OSAM is not.”
- But IMS is strategic and OSAM is an essential part of IMS.
- Therefore, talk of OSAM being “non-strategic” is nonsense

# Features and Benefits of OSAM for Data Bases

---



## ▲ Base Performance

- Specific purpose (rather than general purpose) software
- Buffering
- Syncpoint processing
  - chained writes
  - parallel writes
- “Background Write”

## ▲ OSAM Sequential Buffering

## ▲ OSAM 8GB Data Sets

## ▲ Cached data in CF

# OSAM Base Functions

---



## ▲ Specific Purpose Software

- OSAM has been written for particular IMS usage
  - optimised for these specific functions
- Compare with VSAM, which is a general purpose access method
- Basic Law of Computing
  - *"The more specific the function, the more efficient the process"*
- Benefit
  - reduced CPU cost

# OSAM Blocks and Buffers



## ▲ OSAM Blocksize can be any value up to 32,752

- ▶ VSAM CISIZE is multiple of 512 up to 8K, and multiple of 2K up to 30K

## ▲ There is one OSAM buffer-pool containing any number of subpools

- a subpool is a set (4 to 32,767) of buffers of the same size
- buffer sizes are 512, 1K, 2K or multiple of 2K up to 32K
  - ▶ VSAM buffersizes more than 4K must be multiples of 4K
- any number of sub-pools can have same buffer size
  - ▶ VSAM allows a maximum of 16 (for non-index CIs)
- DB Data Set can be assigned to a specific sub-pool
- subpool parameters specified with IOBF=... in DFSVSMxx

## ▲ Multiple OSAM sub-pools with same size buffers are encouraged

- especially in online systems
- enables more parallelism in IMS buffer management
- reduces buffer-search cost for HD Space Search

**In summary, OSAM offers more flexibility for blocksize, buffersize, and dedicated subpools**

# OSAM Syncpoint Processing

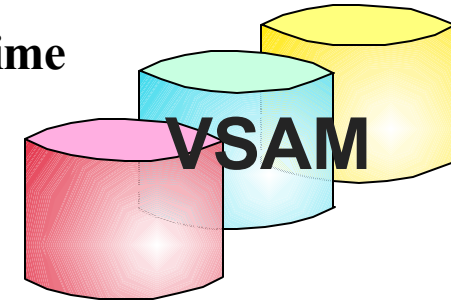


For transactions, BMPs and checkpointing batch programs

→ All IMS DB Writes should take place at Syncpoint time

▲ VSAM writes each buffer individually

■ one at a time ■ ■ ■ ■ ■ ■ ■ ■ ■ ■

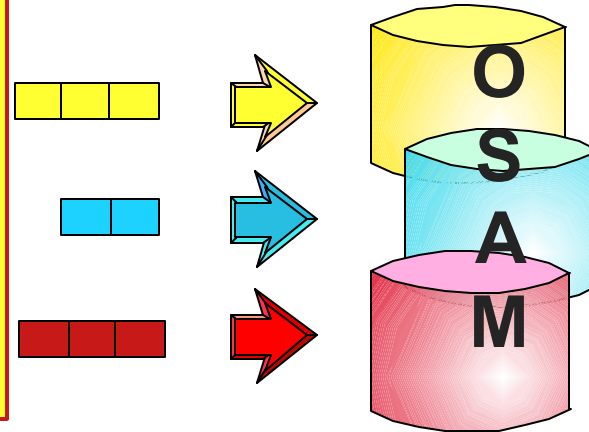


▲ OSAM chains together blocks for same dataset using a single I/O

■ For non-page-fixed subpool, limited to 49 blocks per SIO

▲ OSAM does parallel writes

- for multiple subpools
- to multiple volumes
- and in parallel with VSAM



▲ Benefit

■ Reduced Elapsed Time & Region Occupancy

# OSAM Sequential Buffering

## ▲ Sequential Buffering

### ■ Chained Reads (10 consecutive blocks) instead of single-block reads

- assumes if you need the first block, you will also need the immediately following ones

### ■ Look-ahead reading (asynchronous read-ahead)

- while processing current sequential set of blocks, read the next set
  - ▶ data required by application is always in buffers

## ▲ OSAM Sequential Buffering (OSAM SB)

### ■ 'Enabled' by the user

### ■ Dynamically switched on/off by IMS, according to the estimated/measured benefit

## ▲ Exploited by

### ■ BMPs (and theoretically, MPPs)

### ■ Stand-alone Batch

### ■ Utilities

- Online IC, Unload, Scan, Prefix Update, Surveyor, etc.

## Benefit

- ▶ Totally sequential processes can run in less than a third of the time
- ▶ All jobs with some element of sequential processing will see benefit



# OSAM SB compared with HSSR

---

## ▲ High Speed Sequential Retrieval (HSSR)

- Component of IMS Data Base Tools (DBT)

- Includes an API

  - transparent to programmer

- Supports VSAM and OSAM

- BUT

  - Only for stand-alone batch

  - Only a restricted set of DL/1 calls allowed with HSSR PCB

## ▲ OSAM SB

- totally transparent

- no programming restrictions

- all environments and program types

# OSAM “Background Write”



Prior to IMS V6, the one significant benefit offered by VSAM was Background Write

- For batch jobs (typically, though not exclusively) and especially if running without checkpointing, when all a subpool’s buffers are updated and another read is needed, space must be made in the subpool. This is a “forced write” situation.
- However, with VSAM background write enabled, when IMS notices that a subpool is completely full of altered buffers, a specified percentage of the least recently used buffers would be written out by an asynchronous background lower-priority task
- This prevents the program having to wait for “forced writes to make space”

▲ OSAM “Background Write” introduced in IMS Version 6

▲ The OSAM facility is for stand-alone batch only, and is not a true “background” write

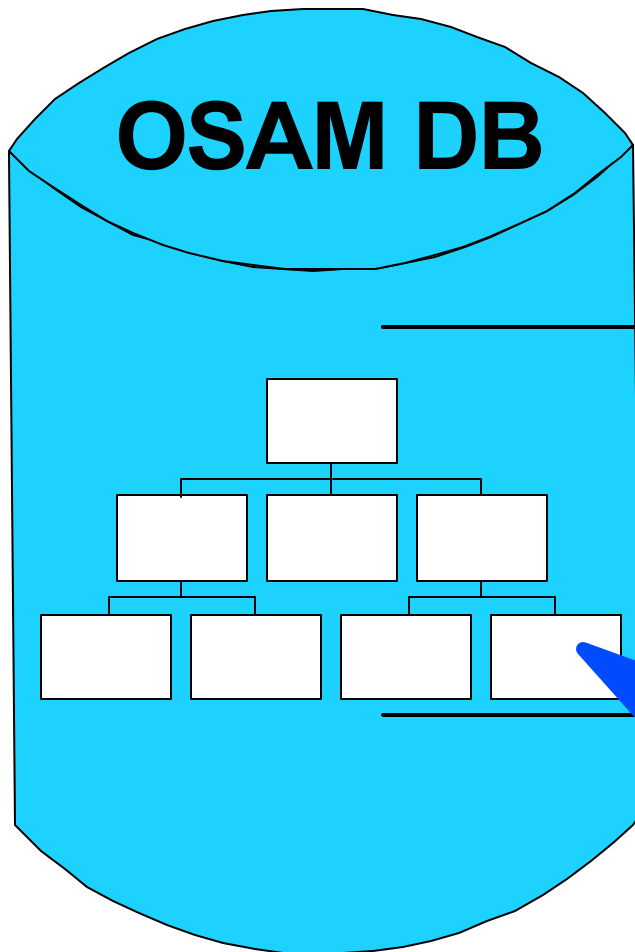
▲ When a forced write for space is needed by a batch program

- OSAM issues a synchronous chained write of all the buffers in the subpool
- Similar to an application CHKP for just the subpool!

# OSAM 8GB Datasets

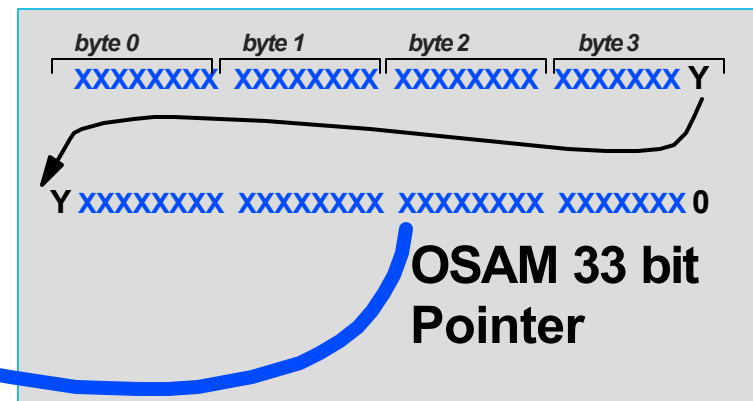


▲ Unlike VSAM which has a 4GB dataset limit, OSAM datasets can be up to 8GB in size

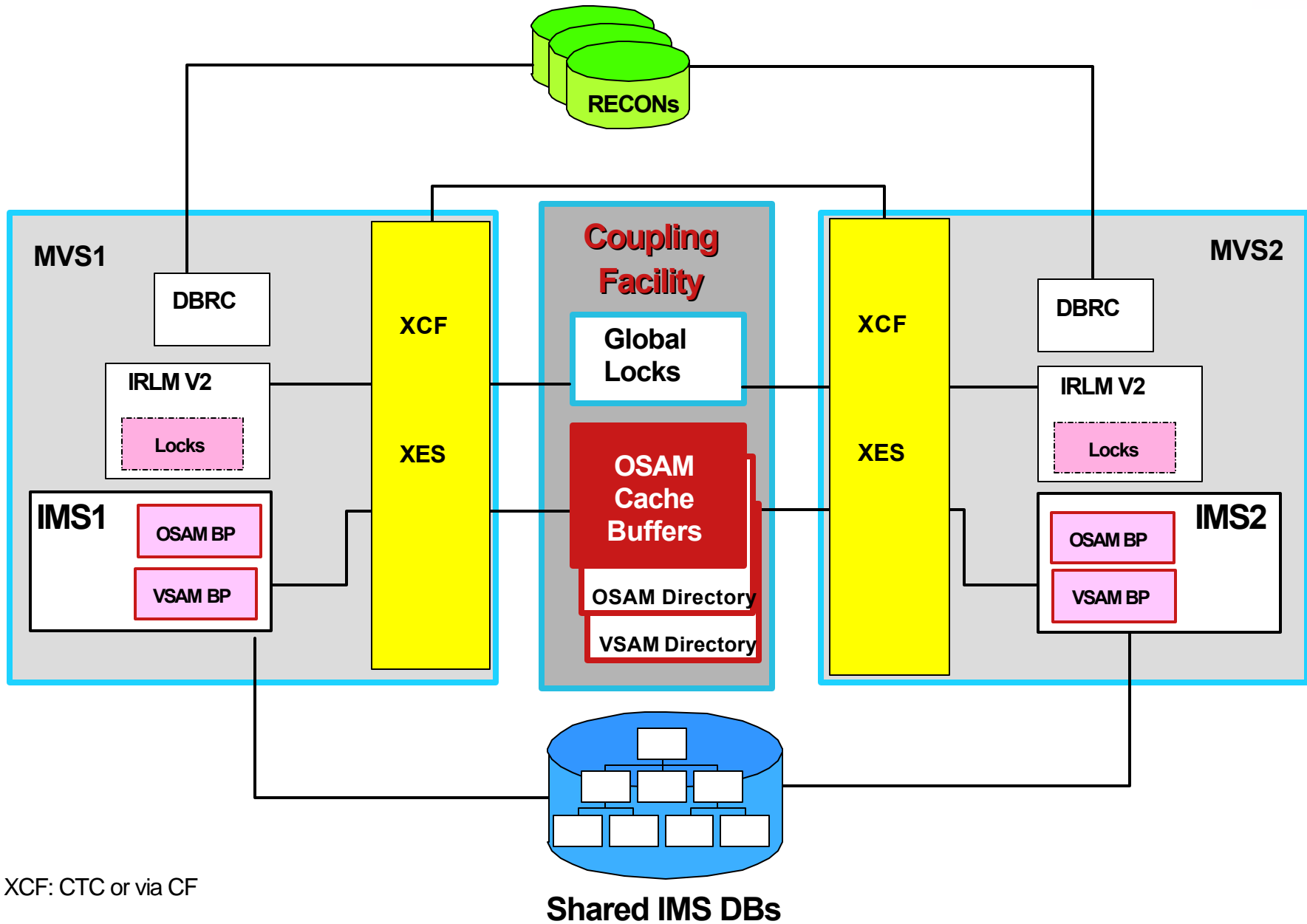


Up to 8 GB

- ★ If blocksize is even, pointer values are always even
  - Rightmost bit is ZERO
  - Can be used as high-order 33rd bit



# Caching OSAM Data in the CF (IMS V6)



XCF: CTC or via CF

# Caching OSAM Data in the CF (IMS V6) ....



## ▲ Objective:

- For small, highly volatile, shared OSAM DBs, reduce impact of OSAM re-read activity due to buffer invalidation
  - replace DASD I/O with CF Access

## ▲ Utilizes store-through cache

- When application reads data from DASD, it is copied (as a user option) into the Coupling Facility
- At application commit, changed data written first to DASD, and then to Coupling Facility
  - before locks released

## ▲ Caching is a user specified option

- Specified at the OSAM subpool level
- IOBF statement in DFSVSMxx
- Choice of cache updated data only or cache all referenced data

# “Every Silver Lining has a Cloud”



- ▲ **Only VSAM allows a subpool to be defined to include buffers in Hiperspace**
  - Hiperspace buffers explicitly exploit “Expanded Storage”, but this is of little relevance these days
  - The main benefit of Hiperspace is when the IMS system often performs “buffer search” as part of the HD Space Search algorithm
    - ▶ VSAM only searches the main (non-Hiperspace) part of the subpool
    - ▶ gives you all the benefits of a very large subpool without the costs of searching through the whole pool
  
- ▲ **OSAM uses JCL to Allocate Datasets rather than AMS**
  - Allocation normally done by DB Load Job
  - But can be pre-allocated if your installation standards require it
    - pre-allocation with multivolume datasets requires care
  
- ▲ **Care is required in *reusing* an OSAM multivolume data set**
  - You potentially could leave an EOF on a volume that is not used on a reload
  
- ▲ **Number of OSAM Secondary Extents is limited**
  - between 52 and 60 (dependent on blocksize)

# Summary



## ▲ OSAM is more efficient than VSAM

- less CPU
- chained writes
- parallel writes
- chained and look-ahead reading with OSAM SB

Reduced Online Region Occupancy  
Reduced Batch Elapsed Times

## ▲ OSAM supports up to 8GB datasets

## ▲ OSAM allows Coupling Facility caching for volatile shared DBs

★ If performance or cost are key factors in your system

➔ **USE OSAM**