# OPS-23: OpenEdge® Performance Basics

Adam Backman
White Star Software
adam@wss.com

## Agenda

- Goals of performance tuning
- Operating system setup
- OpenEdge setup
- Setting OpenEdge parameters
- Tuning APWs
- OpenEdge utilities



## Goals of Performance Tuning

- Consistent performance despite load
- Why not “fast”
  - People want consistency
  - Month end should not be different from the middle of the month
  - People want their “TPS reports” to take the same amount of time each day
  - Fast is relative

## When to Tune Performance?

- Installation or upgrade of hardware
- Installation or upgrade of software
- Change in workload
  - Number of users
  - Volume of data



## Application Performance

The single largest component of performance tuning is application performance.

A well written application will run well given the proper resources while a poorly written application will rarely run well despite resource allocation.



## UPDATE STATISTICS

- If you are a SQL user you should use the UPDATE STATISTICS function within OpenEdge to ensure your queries are as efficient as possible
- How Often? Weekly would be nice but I think monthly is more practical for most
- Caution: This can take some time to run

## Operating System Setup

- Which operating system?
- Network Considerations
- Disks
  - Yes, RAID 5 is still bad.
- Memory
  - Use the memory for the common good first
  - Then get creative
- CPU
  - Good versus bad CPU utilization



## Which Operating System?

Consider the following:

- Your needs and capabilities
  - Uptime requirements
  - Ability to self support
- The ability of the vendor
  - Local support
  - Phone support
  - Quality control
- Price/performance



## Network contention

- The network is the slowest resource for client/server applications so you want to eliminate contention for this resource before moving on to the other resources.
- AppServer™ applications reduce network utilization and improve performance
- Moving large jobs to the server also reduces dependency on the network

## Networking tips

- Keep things local
  - No temp files on network drives
  - Move the application “close” to the user
- Use -cache to speed initial connection
- Use -pls if you are using program libraries over the network
- Application issues are magnified over a network (field-lists, no-lock, indexes, …)



## Disks

- This is where to spend your money
- Goal: Use all disks evenly
- Buy as many physical disks as possible
- RAID 5 is still bad in many cases, improvements have been made but test before you buy as there is a performance wall out there and it is closer with RAID 5



## Causes of Disk I/O

- Database
  - User requests (Usually 90% of total load)
  - Updates (This affects DB, BI and AI)
- Temporary file I/O - Use as a disk utilization leveler
- Operating system - usually minimal provided enough memory is installed
- Other I/O

## Disks - General Rules

- Use RAID 10 (0+1) or Mirroring and Striping for best protection of data with optimal performance for the database
- For the AI and BI RAID 10 still makes sense in most cases. Exception: Single database environments



## File Placement

- Think security of data first and then performance
- The After Image files should always reside in a different location than the database AND before image files
- Best if database and before image can be separated as well



## Data Placement - Example 1

- 16 disks total
- Single database
- 10 disks RAID 10 for the DB
- 2 disks mirrored for OS
- 2 disks mirrored for BI
- 2 disks mirrored for AI

## Data Placement - Example 2

- 16 disks
- Multiple databases (let's say 5)
- 10 disks RAID 10 for DB and BI
- 6 disks RAID 10 for OS and AI



## Memory

- Use memory to avoid disk I/O
- Have enough to eliminate swapping
- Excessive physical page faulting is a good indicator of a memory lean situation
- If you are low on memory get more or reduce utilization (Yes, you paid to see this)



## Memory

- When increasing memory always increase broker parameters first and then increase client parameters
- Buffer hit ratio in promon is a bad indicator of real performance but the goal is to reduce I/O and this ratio shows the "cost" of each request
- If you increase -B buffers you should see a reduction in disk reads at the same workload

## Memory hints

- Increase -B in 10% increments until the point of diminishing returns or swapping, whichever comes first
- Use Post-Version 9 private buffers (-Bp) for reporting
- Use memory for the users closest to the customer first (developers increase last)
- Use -Bt for temp tables - In OE10 they use a minimum of 9 blocks (1 Index + 8 Data)



## CPU

- Buy a single fast CPU or multiple slower ones?
  - A Single fast CPU is good for single threaded operations (nightly report runs)
  - Multi-CPU systems excel at running many small threads at the same time
- Single core processors versus multi-core
  - A dual core is approximately equal to 1.5 single cores (Source: IBM benchmark center)
  - A quad core is approximately equal to 2.75 single cores (Source: IBM benchmark center)



## CPU contention

High CPU activity is not bad in and of itself but high system CPU activity is bad and should be corrected.

A host-based system will have a higher percentage of User time that client/server or n-tier

## Components of CPU activity

- USER - This is what you paid for
- SYSTEM - This is overhead
- WAIT - This is waste
- IDLE - This is nothing ;-)



## CPU activity goals

The goal is to have as much USER time as possible with as little SYSTEM and WAIT.

A practical split is

| | |
|---|---|
| USER: | 70% |
| SYSTEM: | 25% |
| WAIT: | 0% |
| IDLE: | 5% |

This represents a good goal for a host-based system



## Eliminating high SYSTEM CPU activity

- Always use -spin
  - Use a setting of 1 for single CPU systems
  - Use a higher setting for multiple CPU systems
- Testing has shown that the optimal setting will vary significantly based on CPU clock speed
- Start at 2000 and work up (perhaps way up) from there

## Eliminating high WAIT CPU activity

- WAIT = Waiting on I/O
- If you still have IDLE time it generally is not a big problem
- Look at paging/swapping first
- Next look at your disk I/O



## OpenEdge Setup

- Database block size
- Storage Areas
- Before image cluster size
- File placement



## Database Block Size

- Generally, 8k works best for Unix/Linux
- 4k works best for Windows
- Remember to build filesystems with larger block sizes (match if possible)
- There are exceptions so a little testing goes a long way but if in doubt use the above guidelines

## Storage Areas

- Two types (Type I and Type II)
- Allow you to split objects (tables and Indexes) into different physical locations
- Allow you to store more data as you increase the number of areas
- Areas can be split into multiple extents



## Type I Areas

- Data blocks are social
  - They allow data from any table in the area to be stored within a single block
  - Index blocks only contain data for a single index
- Data and index blocks can be tightly interleaved potentially causing scatter



## Type II Storage Areas

- Data is clustered together
- A cluster will only contain records from a single table
- A cluster can contain 8, 64 or 512 blocks
- This helps performance as data scatter is reduced
- Disk arrays have a feature called read-ahead that really improves efficiency with type II areas.

## Storage Areas Compared

| Type I | Type II |
|---|---|
| Data Block | Data Block |
| Data Block | Data Block |
| Index Block | Data Block |
| Data Block | Data Block |
| Data Block | Data Block |
| Data Block | Data Block |
| Index Block | Data Block |
| Index Block | Data Block |
| Data Block | Index Block |
| Index Block | Index Block |



## What is the Before Image Log and How Does is Work?

- Keeps track of data intra-transaction to allow for transaction back out in case of user interaction or system malfunction
- The before image log is made up of clusters which are definable in size
- Transaction notes are written to the log sequentially and blocks can be reused
- It is automatic and cannot be turned off



## How the AI File Works

FOR EACH customer: ← Transaction Begin

UPDATE customer. ← BI Note Written
← AI Note Written

END. ← Transaction End

## Before Image Cluster Size

- This can have a dramatic effect on update performance
- Default is generally too low for most applications
- General rule: 2 minutes between checkpoints at high update times of the day
- This will equate to settings between 1 and 8 MB for most users.

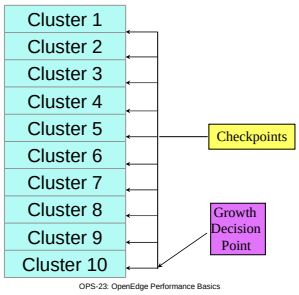


## BI File - Reuse

- Clusters fill sequentially
- When the last formatted cluster is reached there is a reuse decision point
- The “oldest” cluster is examined to determine if it can be reused
- Then, either the oldest cluster is reused or another cluster is added, formatted and used



## BI File - Reuse



## OpenEdge Parameters

- Which parameters are important
- How to set them
- When to watch and change them



## Important OpenEdge Parameters

- -B - Database buffers
- -Bp - Private buffers
- -Mn - Maximum number of servers
- -bibufs - Before image buffers
- -aibufs - After image buffers



## Private Buffers (-Bp)

- Not really private
- Use different LRU chain
- Good for read only clients (reports)
- _MyConnection._MyConn-NumSeqBuffers to allocate or deallocate from the application level

## Tuning APWs

- Start with 1 APW
- Monitor buffers flushed at checkpoint on the activity screen (option 5) in promon
- If buffers flushed increases during the "important" hours of the day add 1 APW



## Tuning Page Writers

- Asynchronous page writer (APW)
- Before image writer (BIW)
- After image write (AIW)



## OpenEdge Monitoring Utilities

- Database Analysis
  - Can be run while the db is up
  - Should be run at least once a quarter
  - How to read it (see example)
- Promon
  - Activity Screen
  - Block Access
- Virtual System Tables (VSTs)
- VST-Based tools (Protop, OE Management, ProMonitor)

## OpenEdge Management Utilities

- Index compact
- Index rebuild
- Table move
- Index move



## Index Compact

- Online reorganization of indexes
- Rebalances B-Tree
- Can increase but not decrease index utilization

```
proutil <db> -C idxcompact [#]
```



## Table and Index Move

- Much faster with type II areas
  - Moving a 2GB index completed in 2 minutes
- Good way to reduce scatter provided you have a fresh area as a target
- Not totally online as the table is locked for the duration of the move

## Index Rebuild

- Offline way to rebuild indexes
- Provides better end indexes due to sorting capability
- Ability to specify packing factor can help avoid block splits

```
proutil <db> -C idxbuild [-pfactor %]
```



## Conclusion

- The application code still has the most effect on performance
- Work from slowest resource to fastest
  Network, Disk, Memory, CPU
- Change one thing at a time to know the effect of changes
- Performance tuning is an ongoing process as things constantly change