By the Pool (with the Kids)

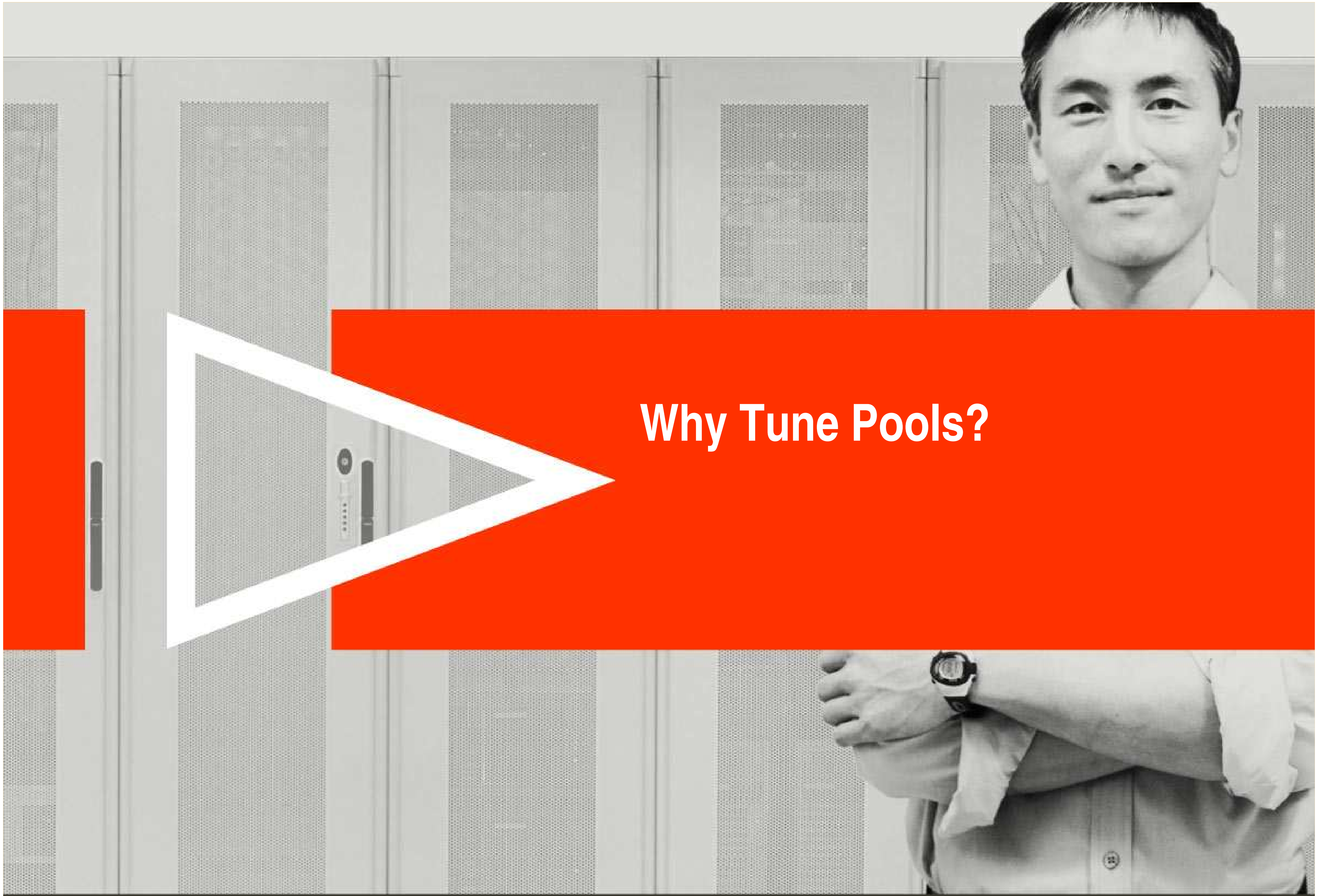
Andy Ward
Principal Software Consultant – BMC Software

Agenda

- The benefits of well tuned buffer pools
- Size isn’t everything
- Useful IFCIDS
- Collecting the data
- Analysing the data

Caveats

- DB2 V7
  - Unless otherwise stated
- Not concentrating on individual methods to collect data
  - Too many monitors and methodologies
- Concentrating on local pools
- This is an overview
  - 45 minutes is too short a time to explore every area

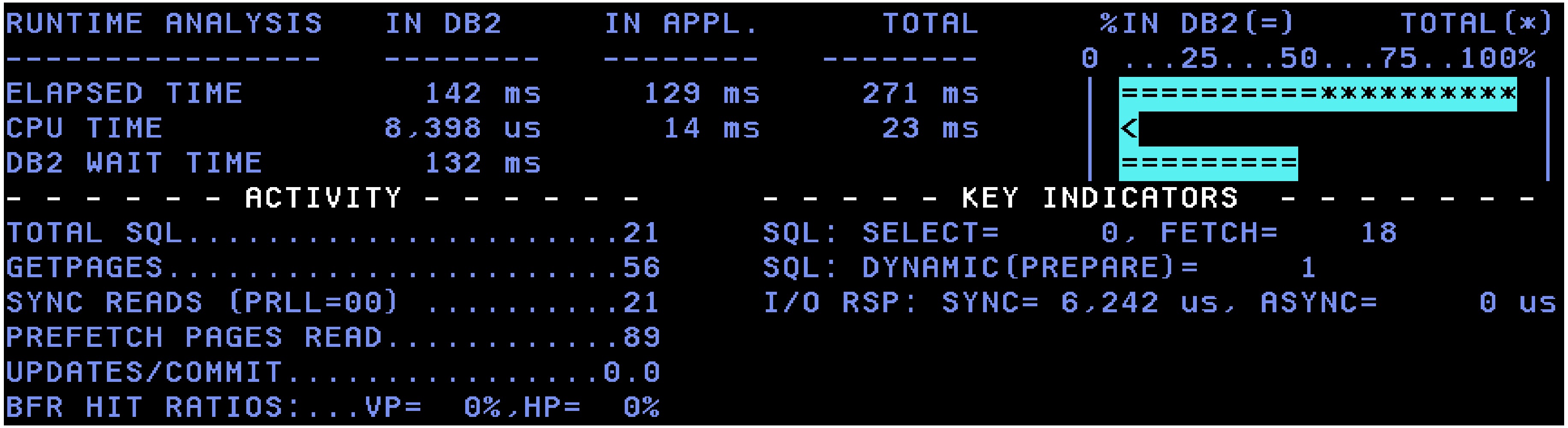


Why Tune Pools?

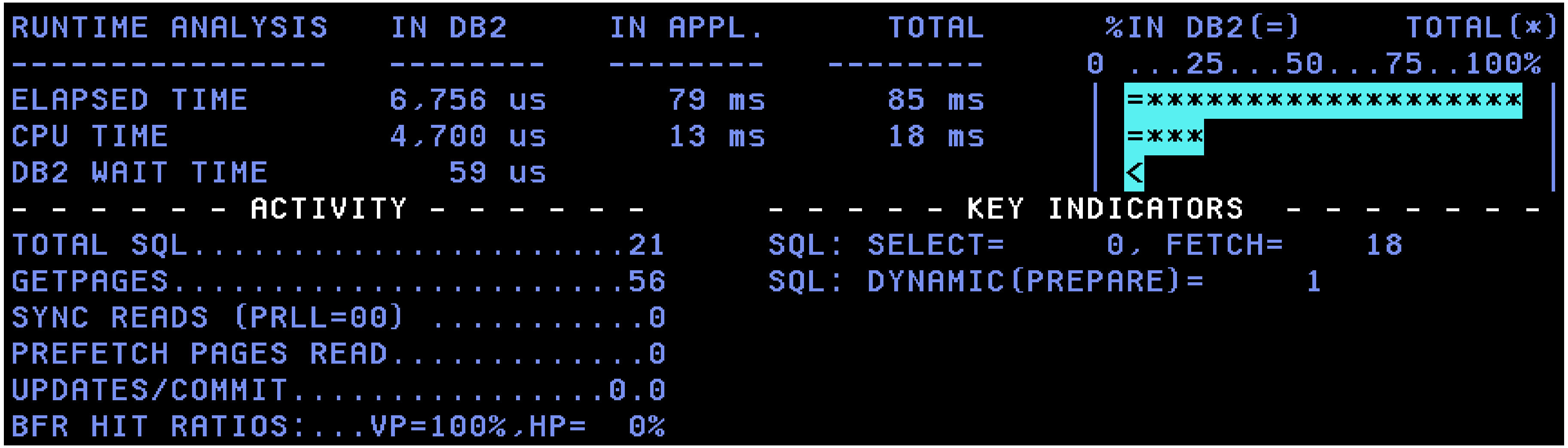
Pool Tuning – The Benefits

- A reduction in IO
  - Hopefully!
- A reduction in IO wait times
  - In turn leading to a reduction in response times and greater throughput
- A reduction in CPU
  - Asynchronous IO charged to DB2
  - Synchronous IO charged to the application
- A potential increase in throughput
- Potentially smaller pools delivering better performance
  - Possible paging reduction

Benefits – The Evidence 1



```
RUNTIME ANALYSIS    IN DB2      IN APPL.       TOTAL       %IN DB2(=)      TOTAL(*)
----------------    ---------   ---------   ---------     0 ...25...50...75..100%
ELAPSED TIME           142 ms      129 ms      271 ms     |==========**********   |
CPU TIME             8,398 us       14 ms       23 ms     |<                      |
DB2 WAIT TIME          132 ms                             |=========              |
- - - - - - ACTIVITY - - - - - -      - - - - - KEY INDICATORS  - - - - - - -
TOTAL SQL......................21     SQL: SELECT=      0, FETCH=     18
GETPAGES.......................56     SQL: DYNAMIC(PREPARE)=      1
SYNC READS (PRLL=00) ..........21     I/O RSP: SYNC= 6,242 us, ASYNC=      0 us
PREFETCH PAGES READ............89
UPDATES/COMMIT................0.0
BFR HIT RATIOS:...VP=  0%,HP=  0%
```



```
RUNTIME ANALYSIS    IN DB2      IN APPL.       TOTAL       %IN DB2(=)      TOTAL(*)
----------------    ---------   ---------   ---------     0 ...25...50...75..100%
ELAPSED TIME         6,756 us       79 ms       85 ms     |=*******************   |
CPU TIME             4,700 us       13 ms       18 ms     |=***                   |
DB2 WAIT TIME           59 us                             |<                      |
- - - - - - ACTIVITY - - - - - -      - - - - - KEY INDICATORS  - - - - - - -
TOTAL SQL......................21     SQL: SELECT=      0, FETCH=     18
GETPAGES.......................56     SQL: DYNAMIC(PREPARE)=      1
SYNC READS (PRLL=00) ...........0
PREFETCH PAGES READ.............0
UPDATES/COMMIT................0.0
BFR HIT RATIOS:...VP=100%,HP=  0%
```

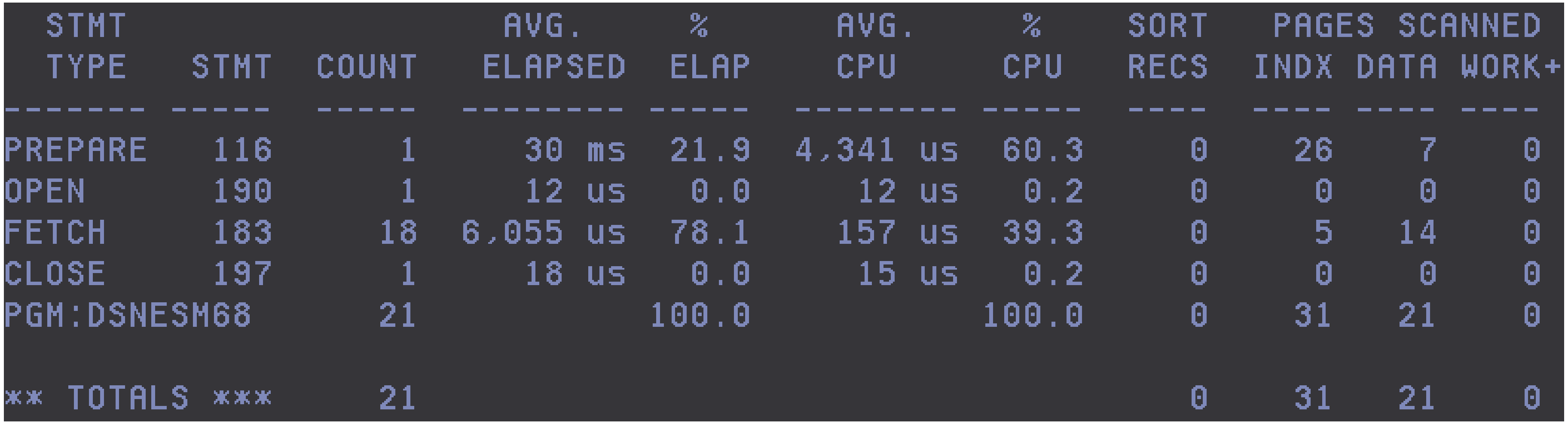
The Wait Time in Real Terms

- Imagine a microsecond (us) equates to 1KM

- You are driving to see a friend
  - Taking the 100% hit ratio example you need to drive 59KM
  - When the hit ratio is 0% you would need to drive 132000KM

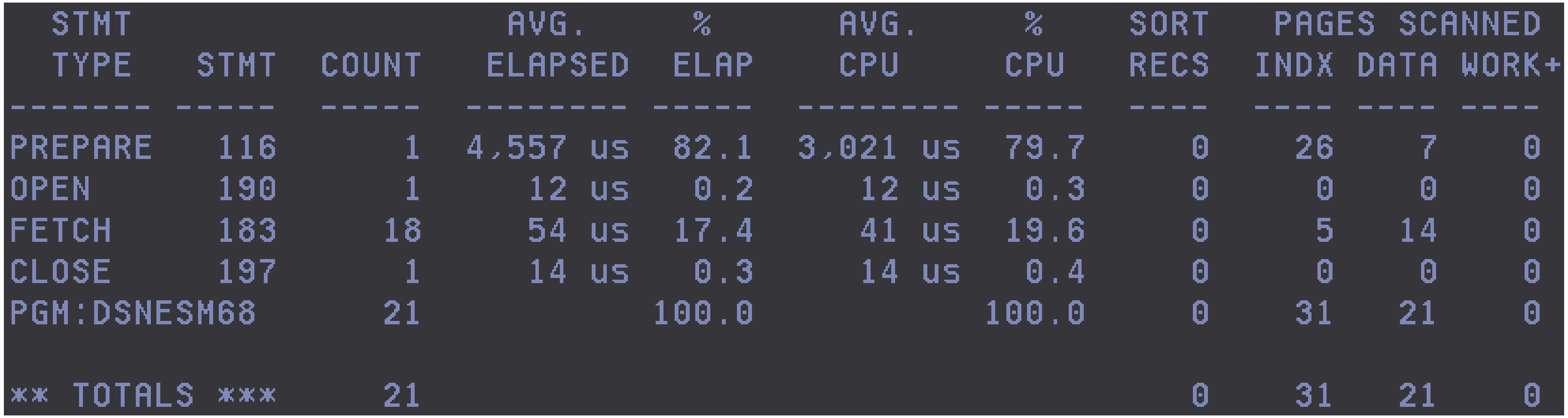
That's over 3 times round the world!

Benefits – The Evidence 2



```
  STMT                    AVG.      %        AVG.       %     SORT    PAGES SCANNED
  TYPE    STMT   COUNT   ELAPSED   ELAP      CPU        CPU   RECS  INDX DATA WORK+
-------- -----   -----  --------- -----   --------- -----   ----  ---- ---- ----
PREPARE    116       1     30 ms  21.9   4,341 us  60.3      0    26     7    0
OPEN       190       1     12 us   0.0      12 us   0.2      0     0     0    0
FETCH      183      18  6,055 us  78.1     157 us  39.3      0     5    14    0
CLOSE      197       1     18 us   0.0      15 us   0.2      0     0     0    0
PGM:DSNESM68        21            100.0             100.0      0    31    21    0

** TOTALS ***       21                                         0    31    21    0
```



```
  STMT                    AVG.      %        AVG.       %     SORT    PAGES SCANNED
  TYPE    STMT   COUNT   ELAPSED   ELAP      CPU        CPU   RECS  INDX DATA WORK+
-------- -----   -----  --------- -----   --------- -----   ----  ---- ---- ----
PREPARE    116       1  4,557 us  82.1   3,021 us  79.7      0    26     7    0
OPEN       190       1     12 us   0.2      12 us   0.3      0     0     0    0
FETCH      183      18     54 us  17.4      41 us  19.6      0     5    14    0
CLOSE      197       1     14 us   0.3      14 us   0.4      0     0     0    0
PGM:DSNESM68        21            100.0             100.0      0    31    21    0

** TOTALS ***       21                                         0    31    21    0
```

The CPU Cost of an I/O

- An excellent presentation contains information on this subject
  - Akira Shibamiya – IDUG 2002 – Session G3
- Using the previous examples
- The average CPU time for a 0% hit ratio was 157us for 18 fetches
  - That equates to 2826us
- The average CPU time for a 100% hit ratio was 41us for 18 fetches
  - That equates to 738us
- Each select executed 21 synchronous reads
  - However the 0% hit ratio select also read 89 prefetch pages

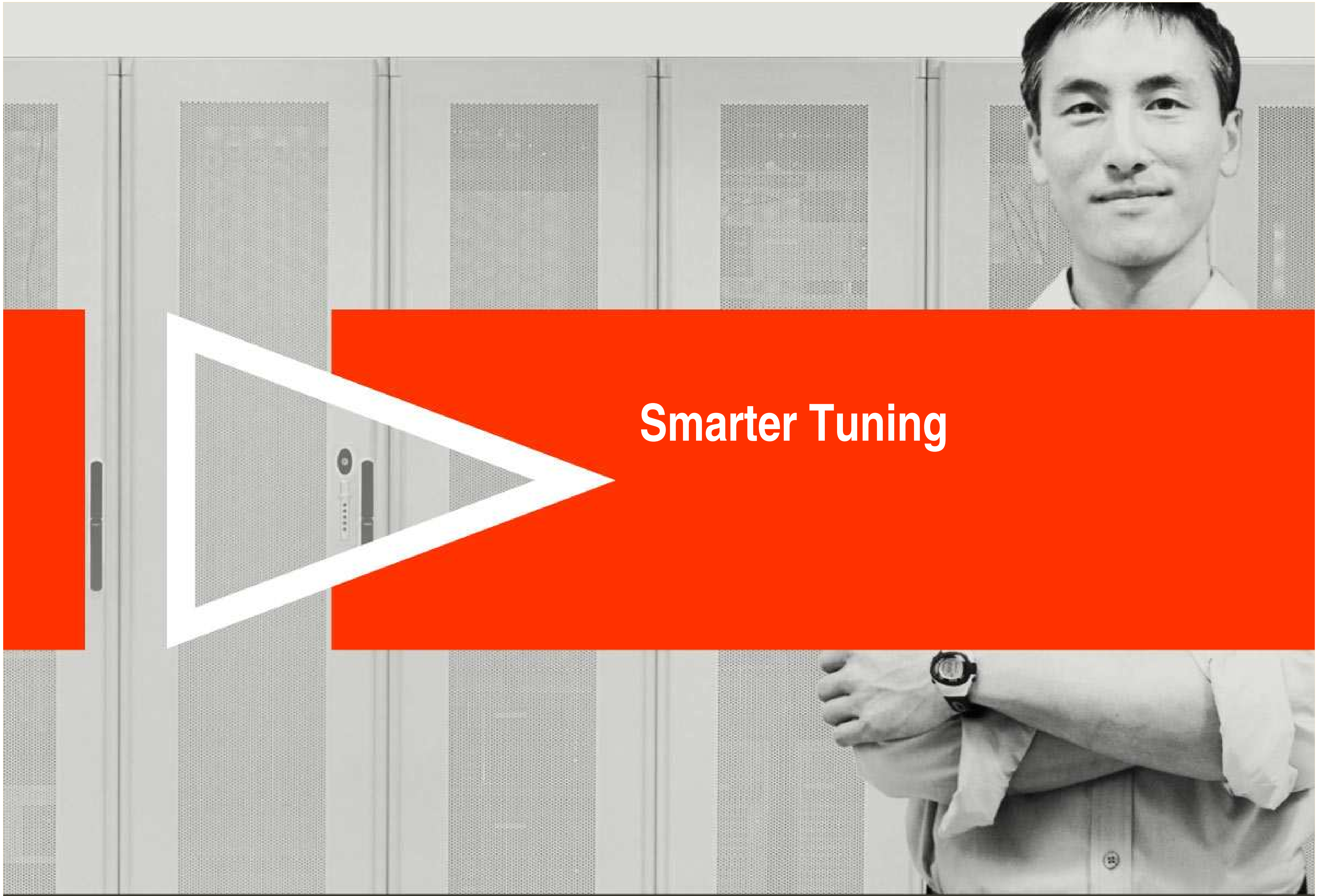
The CPU Cost of an I/O cont’d

- The only difference between the two queries was physical I/O

- Here is the maths…
  - The fetch I/O CPU difference
    - 2826us – 738us = 2088us
  - Minus I/O CPU time for the asynchronous I/O
    - 2088us – (7us * 89) = 1465us
  - Divide this figure by the 21 synchronous I/O’s
    - 1465us / 21 = 69.76us per synchronous I/O

- The accepted figure (z900) is 33us per synchronous I/O (4K page)

- Test this at your shops for a busy transaction and calculate the figure
  - With this information true monetary savings can be calculated



Smarter Tuning

Size Is Not Everything…

- Although it is important
- Other factors critical to well tuned pools
  - Grouping similarly accessed objects together
    - The rest of this presentation will concentrate on how to gather and analyse data to allow you to do this
  - Setting sensible thresholds
  - Collecting valid and pertinent data
    - Don’t just tune your pools for online access
    - Before and after comparison
  - Not taking your eye off the ball
  - Isolate new objects
    - Have development teams provide good CRUD analysis

The DB2 Administration Guide

"You might want to put tables and indexes that are updated frequently into a buffer pool with different characteristics from those that are frequently accessed but infrequently updated."

- **So why not expand on this?**
  - Become more granular in object placement
  - Isolate
    - Large and small objects
    - Randomly accessed objects
    - Sequentially accessed objects
    - Heavily updated objects
    - Indexes and Tablespaces
    - Combinations of the above
- **IBM certainly give us enough pools to do this**
  - But how do I analyse access patterns?

DSNWMSGS

› **Member found in hilvl.DSNSAMP**

- Contains details of IFCID content
- Some very useful pool tuning information
- Information on how to load description data into DB2 tables for easy access

Useful IFCIDS

- 199 – Buffer pool dataset statistics
  - Monitor trace or Statistics class 8
  - Same information as displayed with –DIS BP LSTATS command
  - Interval controlled by ZPARM DSSTIME (default 5 mins.)

- 6 – Beginning of a read I/O operation
  - Monitor trace or Performance class 4
  - Details pool and type of I/O

- 7 – End of read I/O operation
  - Monitor trace or Performance class 4
  - Number of pages read, can be 0 (100% hit ratio)

Useful IFCIDS cont'd

- **8 – Beginning of a synchronous write**
  - These should be avoided at all costs
  - Usually indicates IWTH (97.5% in use pages) has been exceeded

- **10 – Start of an asynchronous write**
  - For both IFCID 8 & 10 you can collect IFCID 9 (write completion) for completeness if required

- **3 - DB2 accounting record**
  - A host of elapsed and CPU time thread information

- **2 – DB2 Statistics record**
  - Accumulated values since DB2 start time
  - Buffer Manager data section
  - Interval controlled by ZPARM STATIME (default 30 mins.)

Useful IFCIDS cont'd

- 198
  - Exceptionally useful for pool tuning
  - Not associated with any trace class
    - Need to specifically list it
  - Records every getpage – be wary of overhead
    - Also notes where the getpage was resolved from
  - Good for calculating working set size
    - More on this later

Thresholds

- DMTH
  - 95% full
  - I/O issued for each row retrieved
- IWTH
  - 97.5% full
  - Synchronous writes to log and disk
- VPSEQT
  - Number of buffers available for prefetch
  - Skip sequential problems?
  - Default 80%
- DWQT
  - Default 50%
  - Percentage of in use pages prior to deferred write being initiated

Thresholds

- VDWQT
  - Default 10%
  - Number of in-use pages for a single object prior to DW being initialised
  - Checkpointing!!

What to Collect?

What to Collect?

- In an ideal world ‘everything pertinent’
  - Bufferpools are generally speaking ‘a subsystem wide resource’

- Overhead is a big consideration though
  - If collecting everything is just not practical
    - Concentrate on critical applications first
      - Isolate by plan
    - Decide on the level of your tuning effort
      - More detail, more benefits, more time, more overhead

- For effective tuning before and after statistics are required
  - Simple bufferpool displays can be extremely useful for assessing tuning success

Data Collection Overhead

- **Virtually impossible to give a ball park figure**
  - Overhead dramatically varies depending on throughput, SQL, number of objects, IFCIDS being selected, filtering etc.

- **A monitor trace is preferred**
  - Only a single trace
  - Output to a flat file
  - No SMF/GTF overhead
  - It requires a DB2 monitor or user written program
  - Use class 30-32 to enable specification of only the IFCIDS required

- **If using a monitor trace…**
  - IFCID 3 provides:
    - Field QIFAAIET – accumulated elapsed time for IFI calls
    - Field QIFAAITT – accumulated elapsed CPU for IFI calls

What The IFCIDS Give You

- **IFCID 3 can help post tuning**
  - Doesn't offer the granularity required for effective tuning
  - Should see improvements in wait times, especially I/O
- **IFCID 2 useful subsystem wide figures**
  - Again no granularity
  - Bear in mind the majority of these values are accumulated from DB2 start
  - Good ball park figures
    - Positive tuning should see I/O per getpage (syncIO/Getpages) decreasing
- **IFCID 6**
  - No prefetch I/O if trace restricted by plan or authid
    - However async I/O doesn't generally impact applications
  - Reread percentage
  - Type of I/O's

What The IFCIDS Give You

- IFCID 7
  - In conjunction with IFCID 6 can be used to determine time between rereads, this is useful for page residency time goals
- IFCID 8
  - There should ideally be none of these
  - Cheaper to monitor for them in IFCID 2
    - However IFCID 8 will highlight DBID & OBID which may indicate a problem space
- IFCID 198
  - Probably the most important IFCID for this type of tuning
  - Provides getpage, relpage, BP hit and update information

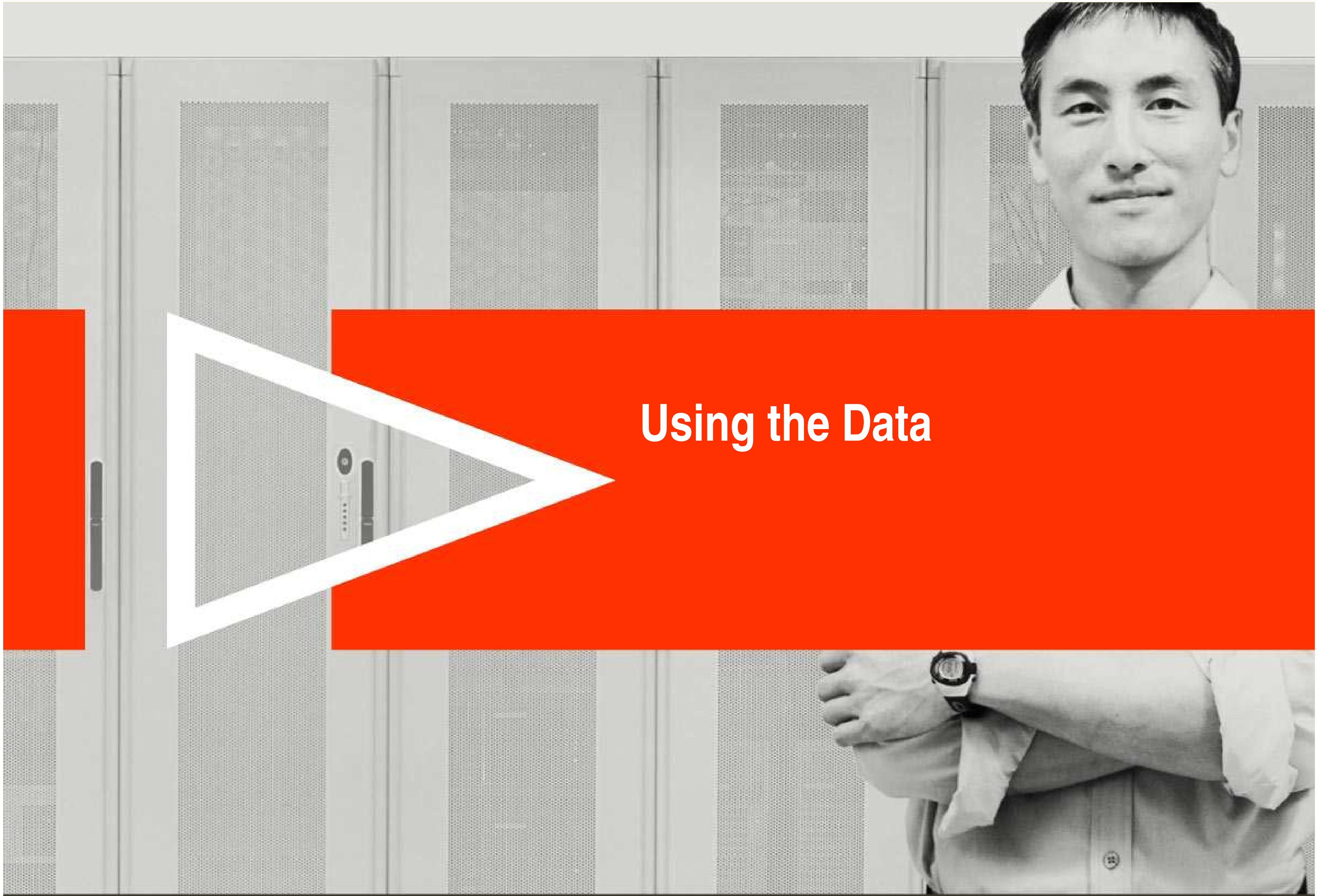
Managing the Collection

- Use trace classes 30-32 and specify only the IFCIDS required
- Define periods of interest
  - Include both online and batch
  - Don’t neglect unusual periods (i.e. month end)
- Collect as much data as possible prior to analysis
  - 5-6 weeks of your chosen intervals is recommended
- Consider sampling
  - i.e. Tracing for 30 seconds every 10 minutes
  - The downside – sampling always relies on extrapolation
- Load the data into DB2 tables for analysis

Hints for Loading the Data

- See IBM Redbook SG24-2244-00 – DB2 for OS/390 Capacity Planning
  - Appendix C – Bufferpool Tuning
  - The book is a little old but the theory is good

- Takes raw DB2 PM report output and loads pertinent data into a table
  - Theory could be applied to any vendors reports



Using the Data

Average Object Working Set Size

- Indicates the amount of buffers required, for a given period, to reduce physical I/O to 0
- More realistic for predominately randomly accessed objects
- High number – object likely to benefit from bring backed by HP
  - In extreme cases own VP and HP
- Use collected IFCID 198 data
- To calculate
  - Select the SUM of a count of the UNIQUE page numbers for a specific object over a time period

Object Access Patterns

- To effectively group objects in separate pools look at
  - Level of sequential access
    - By definition this tells us whether the object is predominately randomly or sequentially accessed
  - General activity levels
  - Update rate
  - Size
- Apply a three tier setting for each of these key indicators
  - High
  - Medium
  - Low

Object Access Patterns cont'd.

- **Gather this information from IFCID 198 records**
  - Load collection interval into a DB2 table
  - Summarize the data into a further table for each interval
    - Total getpages
    - Total sequential requests
    - Total times the page was found in the pool
    - Total updates

- **What's High for getpages and updates?**
  - In relation to YOUR biggest values
    - Analyse YOUR data – an average of the top 10 may be better
      - 33% or less is low
      - 33% - 66% is medium
      - 66%-100% is high

Object Access Patterns cont'd.

- **Calculating**
  - Use the summarised data for a set period
    - Ideally 5-6 weeks
  - Calculate the maximums
    - Either absolute or averages
  - Use case statements to translate numbers into HI, MED and LOW
  - Order by case output
    - This gives groups of objects that would benefit from residing in the same pool with thresholds/sizes set for that specific access

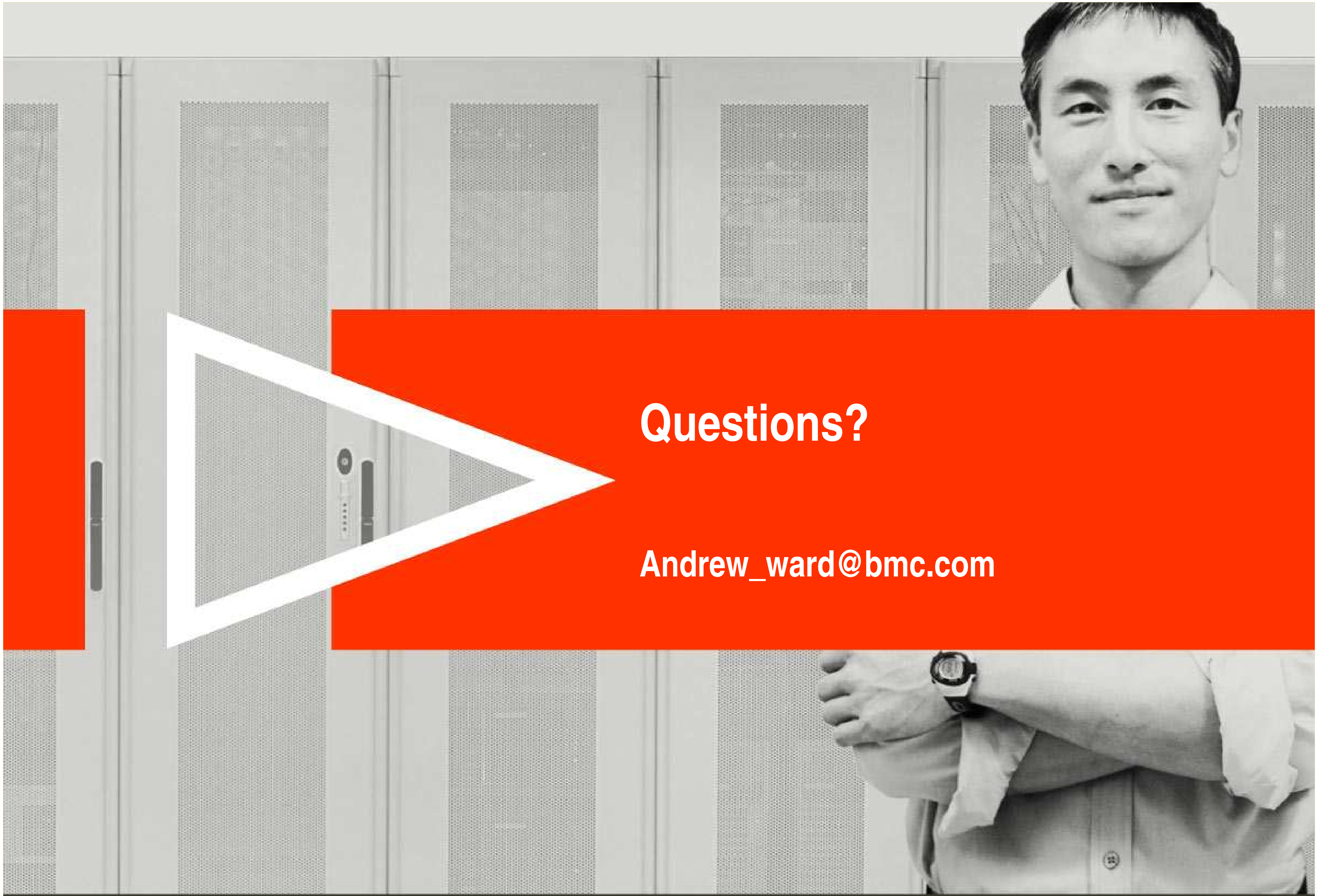
In Summary

A Final Round-Up

- **Smarter Tuning**
  - Aim to group like accessed objects together in their own pools
  - Consider relevant pool thresholds
- **Data Collection**
  - Collect as much pertinent information as overhead will allow
  - Load the data into DB2 tables for ease of reporting
  - Use tools you already own
  - Before and after
- **Using the Data**
  - Find the like accessed objects (analyse IFCID 198 data)
  - Get an idea of bufferpool size requirements, working set size
    - Are Hiperpools required?
- **Finally, Alter the objects, thresholds and size**
  - Don’t forget to reclaim freed up space in existing pools

Speak to Your Vendors

- Tools may be available to help with the task
- Advice on how to use monitors to best effect
  - Which reports show the data required
  - Information/examples of how to load data into tables
- Your company is paying for support and maintenance
  - Get your money’s worth!!!



Questions?

Andrew_ward@bmc.com