



Tuning Techniques To Maximize DB2 Performance

Martin Hubel

Lightyear Consulting

® and ™ indicate USA registration or USA trademark. Other logos and product/trade names are trademarks or registered trademarks of their respective companies.

Copyright©1999, 2000, 2003 crice@lightyr.com 714-438-5391., All rights reserved.

Agenda

- { Objectives and benefits*
- { Types of I/O*
- { Buffer pool tuning methodology*
- { Index tuning*
- { The DB2 tuning project*



Presentation Objectives

- *Describe methods to reduce the cost of DB2 processing for existing applications*
- *Describe ways to reduce physical & logical I/O*
- *Describe the steps necessary to start a DB2 tuning project*



Presentation Background

- *Based on tuning experience with*
 - **EPS: PeopleSoft, SAP and Siebel**
 - **Data warehouses**
 - **Production and test OLTP environments**
- *“Your mileage will vary”*
 - **Tuning opportunities will also vary**
- *Tuning is dependent upon resources available*



The Need For Tuning

- *Use hardware efficiently*
 - Cost of hardware is lower
 - Can you buy enough hardware to solve your performance problem?
- *Productivity of business users and I/T staff*
 - Hardware is cheaper, but people are not
- *Bad performance causes:*
 - Low productivity
 - Need more people to do same work



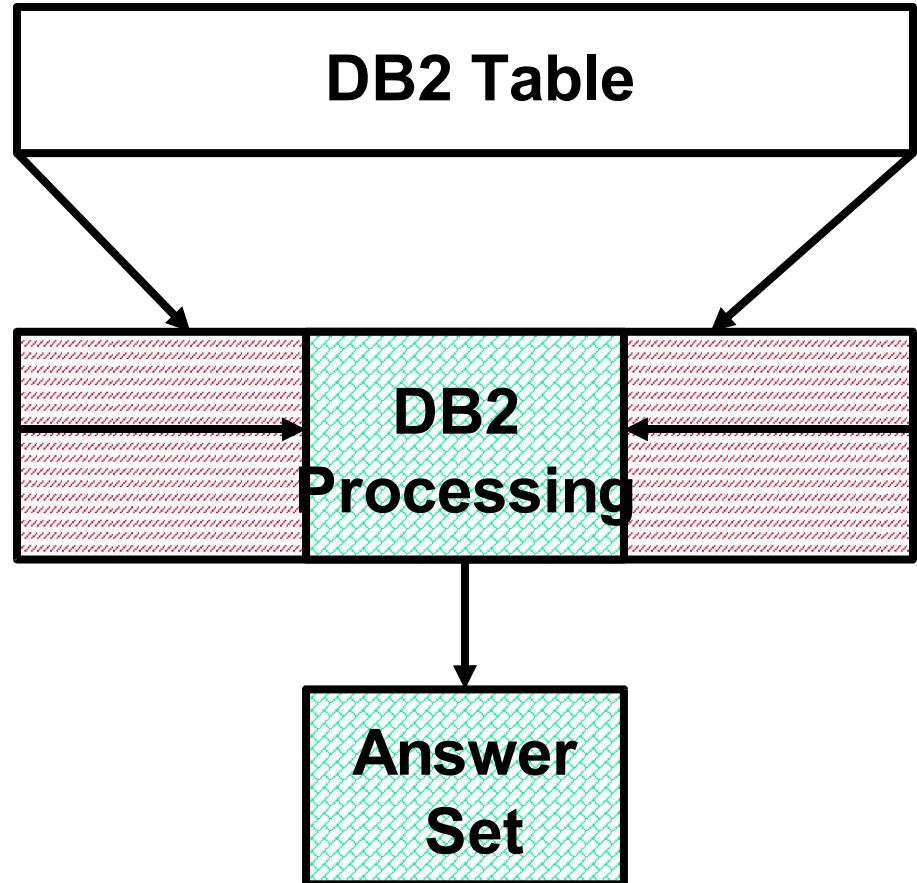
Tuning The Physical Design

- *Almost all physical tuning involves NO changes to SQL text*
 - **>80% of benefits of tuning <5% of SQL**
- *Get physical design in order:*
 - **Buffer pools**
 - **RUNSTATS**
- *Index Changes*
 - **Primary physical design technique for existing applications**
 - **Table and SQL changes not easily done**
 - **May wish to change clustering index**



Logical I/O

- *Answer set size*
 - *Hopefully, answer sets are small*
- *Ask yourself:*
 - **How many rows should be returned from this statement?**
- *Design indexes and SQL to touch less data*



Types of Physical I/O

- *Random*
- *Prefetch*
 - **Asynchronous read ahead capability for table or clustered index scan and utilities**
 - **Pure sequential**
 - **List**
 - **Dynamic**
 - **BIND or execution time decision**



List Prefetch

- *Prefetch for data pages accessed via non-clustered or multiple indexes*
- *32 data pages can be read before first row is returned to application*
 - **Certain "BROWSE" transactions may be adversely affected**
- *Result returned in order as found on DASD*

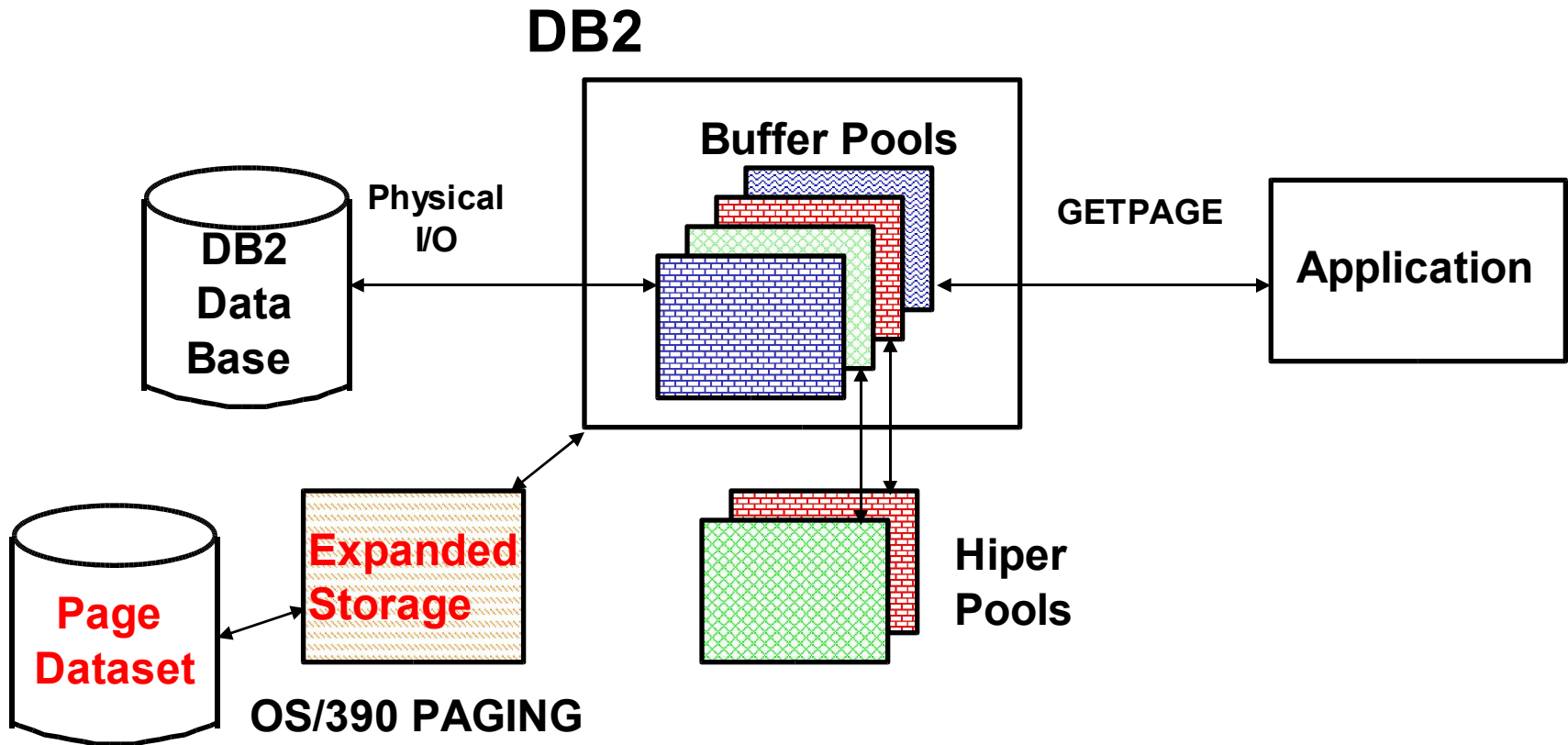


Sequential Detection

- *Also called Dynamic Prefetch*
- *Monitor for sequential pattern in data retrieval at execution*
 - **Turn on if 5 of last 8 pages are within half of the prefetch quantity of each other**
 - **Continue monitor to see if pattern changes back to non-sequential**
- **Max. number of concurrent prefetch engines:**
 - *300 in V5, 600 in V6 [OS/390]*



Buffer Pools and I/O



Buffer Pool Tuning

- *Buffer pools can hold table and index data*
 - **DB2 has fifty 4K buffer pools [OS/390]**
 - **8K (V6), 16K (V6), and 32K have 10 buffer pools each**
- *Concept of logical and physical I/O*
 - **Logical request for data is called a GETPAGE**
 - **GETPAGE satisfied from the buffer pool requires no physical I/O**
 - **Physical I/O causes delays & consumes CPU**



Buffer Pool Thresholds [OS/390]

Fixed Thresholds

Variable Thresholds

*Thresholds
computed as:
(Updated Pages +
Pages In-Use)
BP Size*

IWTH 97.5%

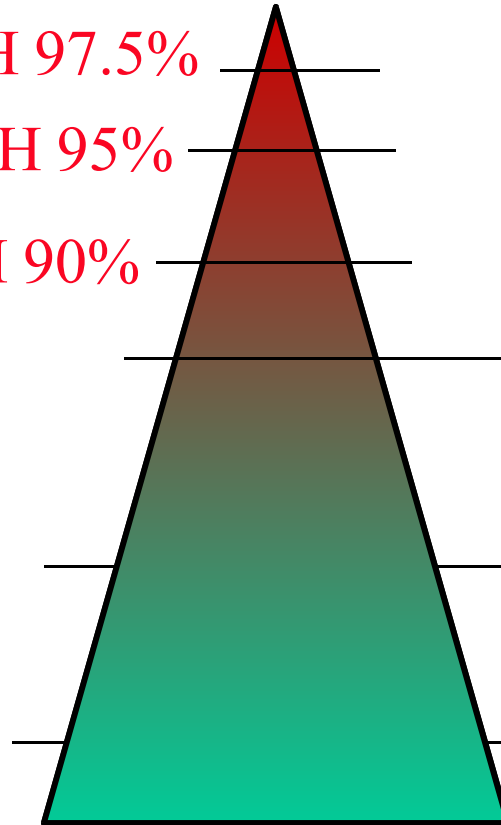
DMTH 95%

SPTH 90%

VPSEQT (80%)
VPPSEQT (50%)

DWQT (50%)

VDWQT (10%)



Setting Variable Thresholds [OS/390]

- *Preferable to use VDWQT rather than DWQT to control asynchronous write activity*
 - On most pools, set VDWQT lower than default 10%
 - Allow a “trickle write” in background
 - Pages written are still in buffer if referenced often
- *IWTH may be non-zero*
 - Not a problem if SPTH and DMTH are both zero
- *DWQT and VDWQT may be set higher for sort/work BP*
- *Set VPSEQT and VPPSEQT lower on random/ mixed pools to avoid dominance by sequential operations*



Hit Ratios

{ *Application hit ratio*

getpages / read I/Os

(getpages-read I/Os) / getpages

- Useful only where almost all random access

{ *System hit ratio is more accurate*

(getpages-pages read) / getpages

- ♥ Asynchronous I/O (prefetch) has negative impact on BP efficiency



Typical BP Starting Points [OS/390]

- *Most people have gotten this far before a tuning exercise is started:*
 - **Catalog and Directory in BP0**
 - **Temporary (work) TS in BP1**
 - **Table spaces in BP2**
 - **Indexes in BP3**
- *Minimum size for any 4K pool should be 1000 pages*
 - **To get maximum prefetch quantity**



Methodology For Tuning

- ***Objective: increase residency of high-use pages in buffer pool***
 - **Gather statistics during key intervals**
- ***Size buffer pools appropriately***
 - **Avoid system paging**
- ***Separate table spaces from indexes***
 - **V6: place LOBs in a separate BP [OS/390]**
- ***Separate by type of I/O***
- ***Use separate BP for high use read-only objects***



Separate Sequential From Random I/O

- *Sequential I/O involves reading a large number of pages to find data that qualifies*
 - **Can push many high-use random pages out of the buffer pool**
- *Best to separate frequently scanned objects from randomly accessed objects*
 - **Use tool to identify number & type of I/O by object**



Key Objects To Isolate

- *Code/reference tables*
 - **Small, high use tables**
- *High use indexes*
- *Large tables*
 - **Isolate table spaces with high sequential access**
 - **Sequential access may indicate index tuning needed**



Very Large Tables

- *Example: atomic table in data warehouse*
 - 7 million pages in 37 partitions
 - High use via index access
 - No identifiable hot spots
- *Small likelihood that desired page will be found in BP*
 - Negatively impact other objects in BP
- *Place table space in a separate BP of 1000-2000 pages*
 - Possibly include other large tables in same BP for this application



Hiper Pools

- *Useful to alleviate DBM1 virtual storage constraint*
 - Up to 8GB total
- *I/O saving when page is not in VP but is in HP*
 - Caches only clean pages
 - Page must be moved to HP from VP when not in VP
- *Recommend HPsize > VP size*
 - Aim for HP R/W ratio (HP Reads/Writes) > 10%
 - HP Read/Write failures indicate ES shortage
- *Maximize central storage before using expanded*



Recent Enhancements

- *New page sizes (v6)*
 - **8K and 16K**
 - 10 buffer pools for each
 - **Logical pages per write I/O**
 - Write I/O is 128K
 - Divide Page size into 128 to find #pages/writeIO
- *Default buffer pool parameters in DSNZPARM (v6)*
 - **Catalog and directory in BP0 (as before)**
 - **Set different defaults BP for TS and IX**



V6 Enhancements

{ *Improved write threshold for VDWQT*

- ♥ For very large pools: > 20,000 pages

- ♥ VDWQT(0,100) □ use 100 buffers for VDWQT

{ *Buffer pool page stealing option*

- ♥ LRU or FIFO

- ♥ Use FIFO only when little or no I/O or when virtually no chance of page being reused in BP

{ *LOB table spaces*

- ♥ Place in separate small buffer pool



V6-V7 Buffer Pools in Dataspaces

{ *Expand beyond current limit of 1.6 GB for VBP in DBM1*

♥ Max dspace size is 8 million pages (any page size)

♥ Can exploit large memory on processors before 64-bit

{ *Data spaces can reside in central or expanded storage*

♥ Specify VPTYPE(DATASPACE) - I/O directly against dataspace

♥ Hiperpools cannot be used with data spaces

♥ Shared lookaside buffer in DBM1 for each page size

┆ Expanded/contracted based on usage

{ *Dataspace Buffer Pool is recommended over*



Copyright©1999, 2000, 2003 Martin Hubel Consulting Inc., All rights reserved.

hiperpool for Z Series processor with OS/390 V2R10

DB2 for z/OS V8 64-Bit Support

{ Move above the bar

♥ 24 bit: 16MB – below the line

♥ 31 bit: 2GB – above the line

♥ 64 bit: 8B times bigger – above the bar

{ Virtual storage management much simpler

{ No dataspace or hiperpools – not needed

{ Concerns shift to real storage



Indexes

- { *Faster access to data*
- { *Enforce primary keys in DB2 referential integrity*
 - ♥ Consider index on each foreign key / alternate key
 - ♥ Ensure uniqueness of values in non-key columns
- { *Partitioned table spaces require clustering index*
- { *A table can have multiple indexes*
- { *Indexes can be added or dropped at any time (e.g. add indexes for special time of year processing)*



DB2 Index Structure

{ *Indexes can have up to 64 columns*

- ♥ **Uses as many columns as possible to delimit index scans**
- ♥ **Can use inequalities, e.g. >, >=, <, <=**
- ♥ **Can use index columns to avoid sort**

{ *Index clustering*

- ♥ **Restored by REORG**
- ♥ **Statistic collected by RUNSTATS**



Columns Which Should be Indexed

- *Columns frequently used in WHERE clauses*
 - Columns that are not frequently updated
 - Column must have sufficient cardinality to be a viable choice for DB2 Optimizer
- *Sorts*
 - Column specification can be ASC or DESC
 - Avoids sort in index ordering is used
- *Definition of Cardinality:*
 - Tables: Number of rows in table
 - Indexes: Number of distinct values in index
 - Unique index cardinality matches table cardinality

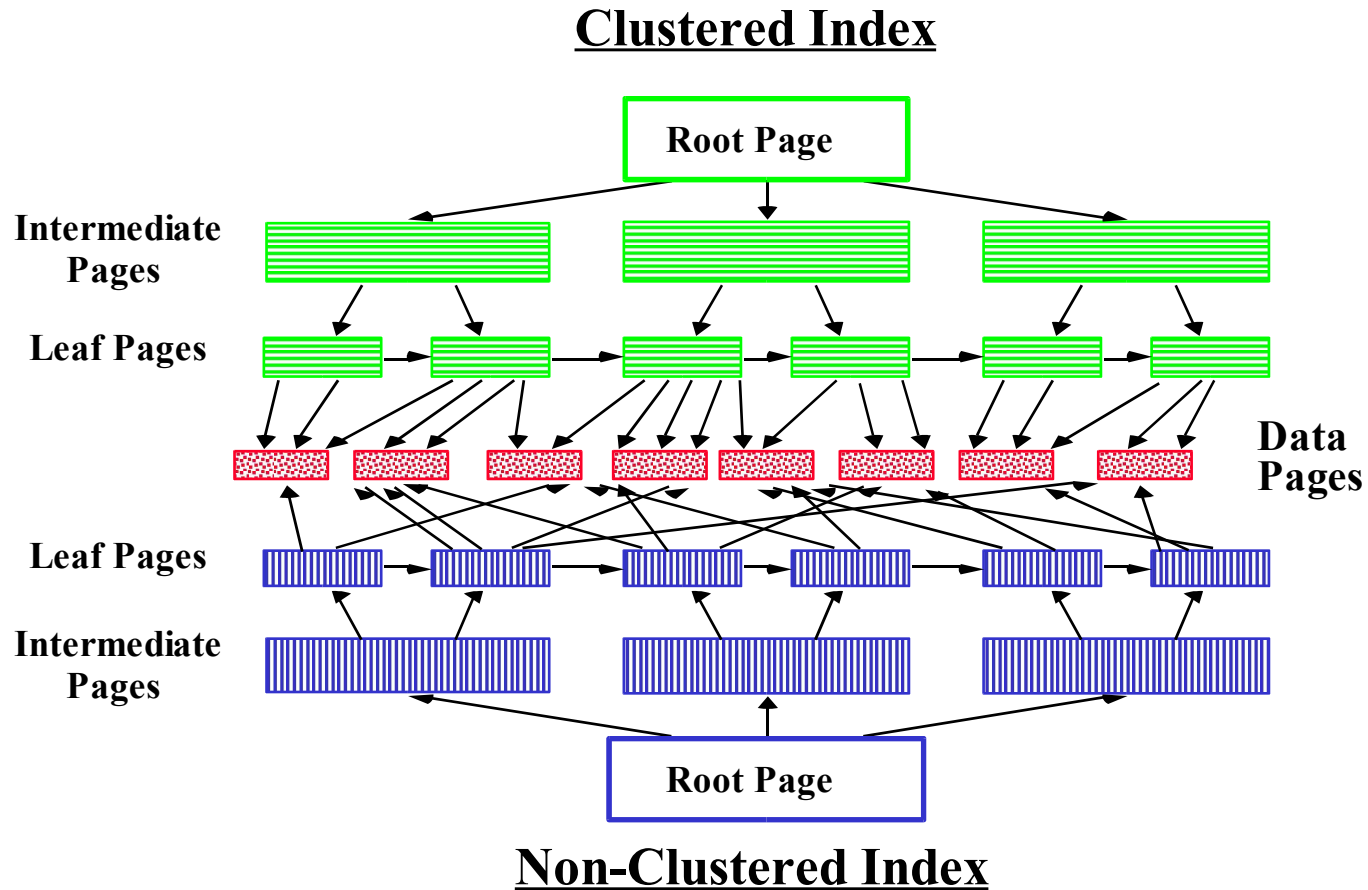


Clustering Index

- *Clustering index stores data in a table in the order of the index*
 - This index is best for retrieving multiple rows
 - More qualified values found per page
- *Clustering: defined as CLUSTER*
 - Clustered describes physical state
 - Clusterratio, Clusterfactor
 - Respected by inserts
 - Restored by REORG



Clustering Order



Disadvantages of Indexes

- *Increases storage or disk space*
- *Increases overhead for update activity*
 - **Each insert and delete causes index update**
 - **Updates to indexed columns**
- *Each index requires its own dataset in OS/390*
- *Locking problems more likely with indexes if concurrent access*



Index Tuning Myths

- *Many redundant indexes*
 - **“Performance problem? Add this index”**
 - Results in more indexes to update
 - Possible buffer pool contention
- *Columns added to achieve:*
 - **Index keys unique or with higher cardinality**
 - **Index only access**
 - **Fewer index entries per page**
 - **More index I/O**



Index Change Guidelines

- 1. Do not change the primary index*
- 2. Expect little index only access*
- 3. Check clustering index placement*
- 4. Remove redundant indexes*
- 5. Remove unnecessary index columns*
- 6. Add index columns if needed*
- 7. Add indexes carefully when necessary*



Comparing Table and Index Statistics

```
SELECT I.NAME, NPAGES, CARD, FIRSTKEYCARD AS FIRSTK, FULLKEYCARD AS FULLKEY, NLEAF,
       NLEVELS AS NLEV, CLUSTERING || CLUSTERED AS CL, UNIQUERULE AS U, T.COLCOUNT AS TBCOL,
       I.COLCOUNT AS IXCOL, RECLENGTH AS RECLEN FROM SYSIBM.SYSTABLES T, SYSIBM.SYSINDEXES I
WHERE  T.CREATOR = I.TBCREATOR AND T.CREATOR = 'F6' AND T.NAME = I.TBNAME AND CARD >20000
ORDER BY CARD DESC, 1
```

NAME	NPAGES	CARD	FIRSTK	FULLKEY	NLEAF	NLEV	CL	U	TBCOL	IXCOL	RECLEN
PS_JRNL_LN	132381	6284688	6	6284688	77261	4	YY	U	28	6	199
PSAJRNL_LN	132381	6284688	1712	2357	11740	3	NN	D	28	3	199
PSBJRNL_LN	132381	6284688	36110	6282385	58064	4	NY	D	28	4	199
PSCJRNL_LN	132381	6284688	1	50442	9061	3	NN	D	28	4	199
PSDJRNL_LN	132381	6284688	721	1183	11534	3	NY	D	28	2	199
PS_DEPRECIATION	61952	3647937	4	3647937	93694	4	YY	U	24	16	137
PSADEPRECIATION	61952	3647937	33	242	5794	3	NN	D	24	3	137
PSBDEPRECIATION	61952	3647937	4	156613	7478	3	NY	D	24	5	137
PS_VCHR_ACCTG_LINE	85189	2679644	7	2679644	39694	4	YY	U	102	9	649
PSAVCHR_ACCTG_LINE	85189	2679644	3	3117	4310	3	NN	D	102	5	649
PSBVCHR_ACCTG_LINE	85189	2679644	317	317	4092	3	NN	D	102	1	649



Cardinality Exercise

- *Objective: Using simple calculations, determine whether if indexes will improve performance*
- *Table: 500,000 rows on 24,000 pages*
- *What is the minimum usable cardinality for a non-unique, non-clustered index?*
- *Consider:*
 - **Number of prefetch I/Os (prefetch quantity of 32)**
 - **Versus**
 - **Number of index and data page I/Os**



Cardinality Calculations

- ***Prefetch I/Os to read whole table***
 - **$24,000 / 32 = 750$ prefetch I/Os**
- ***Non-clustered :***
 - **Index keys/RIDs located**
 - **Reverse solving for data I/O of 750 pages read: $500,000 / 750 = 667$ minimum fullkeycard**
 - **Reality: be suspicious of any index with $< 2,000$ fullkeycard**
 - **Verify results with Explain**
- ***100% Clustered index***
 - **Rows per page: $500,000 / 24,000 = 20$ RPP**
 - **20 times fewer I/O on full key**



Hardware Data Compression

- *DB2 hardware data compression*
 - Fewer pages can mean fewer I/Os
 - Table spaces only, not indexes
 - Compression is at the row level
- *Effectiveness depends upon data patterns*
 - Use DSN1COMP to evaluate
- *Savings vary between applications*
 - Very good upside with little downside risk



Implementing Hardware Data Compression

- *Sample Compression*

DSNURWT - COMPRESSION REPORT FOR TABLE SPACE F6GL.GLJOURLN

688702 KB WITHOUT COMPRESSION

265843 KB WITH COMPRESSION

61 PERCENT OF THE BYTES SAVED FROM COMPRESSED DATA
ROWS

100 PERCENT OF THE LOADED ROWS WERE COMPRESSED

188414 PAGES REQUIRED WITHOUT COMPRESSION

74645 PAGES REQUIRED WITH COMPRESSION

60 PERCENT OF THE DB2 DATA PAGES SAVED USING COMPRESSED DATA

- *Saved 50% CPU on one table*

Copyright©1999, 2000, 2003 Martin Hubel Consulting Inc., All rights reserved.

Do not expect the same improvement across the

Physical Data Organization

- *Data in poor physical condition*
 - Requires more getpages and physical I/O
 - Decreases buffer pool efficiency
 - Increases CPU usage
- *Schedule regular reorganizations for table spaces and indexes*
 - Automate by threshold



The DB2 Tuning Project

- *Driven by business need*
 - Improve customer service
 - Avoid hardware upgrade
 - Reduce charge back costs
- *Often funded by business unit or application development*
 - Tied to business benefit



The Project Team

- *Requires skills for several areas*
 - **DBA**
 - **Systems programming / capacity planning**
 - **Application development**
- *Most resources are internal*
- *Outside consulting help*
 - **Methodology and specific tuning expertise**
 - **Objectivity and viewpoint**



Materials Needed

- *Hardware and software configuration*
 - **WLM Information**
 - **DSNZPARM**
 - **IRLM parameters**
 - **CICS attachment (RCT) parameters**
- *Catalog information*
 - **Physical DB2 design**
 - **Access to tool for reviewing database statistics**
 - **Historical statistics, if available**



Materials Needed

- ***Performance Monitor and System Configuration***
 - **Monthly subsystem statistics**
 - **Monthly accounting summary, or several typical days**
 - **Access to on-line monitor for reviewing DB2 system configuration**
- ***Key programs / transactions***
 - **Code of one or two key programs to see the quality of SQL coding**
 - **SQL access patterns**



Summary

- *DB2 tuning techniques typically yield 30% improvement or more*
- *First try techniques that avoid application coding changes*
 - **Buffer pools**
 - **Index redesign**
 - **Other areas such as data compression**

