Applications – Who needs them??

Tom Renz
August 2001

Introduction

- What is an application?
- Why do we need an application?
- How do we use an application?
- When do we use an application?
- Where do we use an application?
- Who uses/relies on an application?

What is an application?

- A program
- A job
- A command file
- A subroutine
- A storage area of data
- A combination of one or more of the above together to accomplish a given task/outcome/result/etc.

August 2001 COTC - T.RenzSoft 3

Why do we need an application?

- Processes our paychecks, ordering, manufacturing, airline tickets, etc.
- Provides jobs within the department, region, and entire company
- Provides jobs and new ideas for outside vendors

August 2001 COTC - T.RenzSoft 4

How do we use an application?

- Utilize some form of a programming language (3GL, 4GL, CI Commands, etc.)
- Execute a set of commands or processes to provide a desired outcome

August 2001 COTC - T.RenzSoft 5

When do we use an application?

- Run on a periodic basis (hourly, daily, weekly, monthly, etc.)
- Submit via a schedule, other software, via a check list, etc.
- When we require the outcome of the application for other decision making

August 2001 COTC - T.RenzSoft 6

Where do we use an application?

- On our HPe3000
- On our PC
- Via the Web
- Via another computer system utilizing some form of a Client/Server process
- Etc….

August 2001 COTC - T.RenzSoft 7

Who uses/relies on an application?

- Company employees
- Current and Potential customers
- OS providers
- Third party software providers
- Hardware providers
- Database tool providers
- Performance tools and surveys
- Etc……..

August 2001 COTC - T.RenzSoft 8


"....just throw more hardware at the problem."
....a previous Manager in charge of budgets. (1990)
August 2001
COTC - T.RenzSoft
9


"I don't need to be concerned with performance, I'm using Speedware and it will take care of it."
....a Consultant (1996)
August 2001
COTC - T.RenzSoft
10


"...but it worked fast in test."
....lots of development staff (ongoing)
August 2001
COTC - T.RenzSoft
11

"If you take care of the machine, it will take care of you!"

....Tom Renz
(Since 1983)

August 2001 COTC - T.RenzSoft 12

Tools and Techniques to improve your application

- Program recommendations
- File recommendations
- Speedware recommendations
- Suprtool recommendations
- Before and after timings

August 2001 COTC - T.RenzSoft 13

Program recommendations

- Use and abuse of tables
- Counters
- Reference modification (COBOL)
- In-line PERFORMs (COBOL)
- GoTo statement
- Functions
- Other suggestions

August 2001 COTC - T.RenzSoft 14

Use and abuse of tables

Abuse:

- $Control BOUNDS – adds extra code
- Initialize each field within each row
- Processes/checks every defined row and field no matter how many entries in table
- Use of poor index field definitions

Use and abuse of tables (continued)

Ways to improve table processing:

- Use INITIALIZE “function”

```
INITIALIZE TABLE-A.
```

- Move LOW-VALUES when initializing a table of counters
- Initialize first main row, then use this row to initialize remaining rows

```
PERFORM VARYING A FROM 2 BY 1
        UNTIL A > MAX-TABLE-SIZE
    MOVE TABLE-ROW(1) TO TABLE-ROW(A)
END-PERFORM.
```


Use and abuse of tables (continued)

More ways to improve tables:

- Keep and check for highest record in table – don’t read entire table during each pass
- Example:
  - 3rd-party application
  - 3-dimensional table w/ 200 by 300 by 400 rows
  - during program run went to the limits for each row
  - took around 8+ hours to run each month
  - modified to keep a counter for each row and process only to the current limit
  - now takes 15 – 20 minutes to run each month

Use and abuse of tables (continued)

And other ways to improve tables:

- Use of poor index field definitions
- Poor choices (why?):
  - Misaligned on non-word boundaries
  - ASCII/Display/Packed number fields
  - Unsigned integer fields
- Good choices:
  - Double Integer fields (4 bytes – 2 bytes also ok)
  - Signed integer fields
  - Aligned on word boundaries

Counters

Good and bad counter definitions:

- Same as Index Field definitions
- Poor choices (why?):
  - Misaligned on non-word boundaries
  - ASCII/Display/Packed number fields
  - Unsigned integer fields
- Good choices:
  - Double Integer fields (4 bytes – 2 bytes also ok)
  - Signed integer fields
  - Aligned on word boundaries

Counters - Timings

- Unsigned counter timings*:
  CPU: 0.471 ms; Wall: 0.475 ms
- Signed counter timings*:
  CPU: 0.408 ms; Wall: 0.415 ms
- Display counter timings*:
  CPU: 0.795 ms; Wall: 0.802 ms
- Misaligned counter timings*:
  CPU: 0.795 ms; Wall: 0.803 ms

*Based on loop that adds 1 to a counter defined in various formats (signed integer, display and misaligned for 1–million times. (917LX)

Reference modification (COBOL)

- aka - Byte referencing
- Speed of memory instructions
- Old method – created table array for each byte in a field

```
01 FIELD-A   PIC X(80).
01 FILL REDEFINES FIELD-A.
    05  BYTE-A   PIC X  OCCURS 80.
.......
MOVE '   ABC' TO FIELD-A.
                ..or..
MOVE 'A' TO BYTE-A(4).
MOVE 'B' TO BYTE-A(5).
MOVE 'C' TO BYTE-A(6).
```

- New method – use new feature

```
MOVE "ABC" TO FIELD-A(FROM-BYTE:END-LENGTH).
```

August 2001 COTC - T.RenzSoft 21

Reference modification - Timings

Old method timings*:

- CPU: 57.087; Wall: 57.580 (unopt)
- CPU: 52.262; Wall: 52.723 (opt)

New method timings*:

- CPU: 5.429; Wall: 5.477 (unopt)
- CPU: 5.367; Wall: 5.416 (opt)

*Based on loop that moves a string to a byte location and space fill the remaining bytes to the end of the string for 1–million times. (Move 'DEF' to Field-A(7:) (917LX)

August 2001 COTC - T.RenzSoft 22

In-line PERFORMs (COBOL)

- Similar to "DO" Loops in Fortran, "FOR" Loops in Basic, "LOOP" in Speedware, etc.
- Faster, more efficient and less object code for branching instructions to a separate paragraph

August 2001 COTC - T.RenzSoft 23

In-line PERFORMs - example

Left justify data in a string:

```
01 Byte          Pic s9(9)  Comp  Value 0.
01 Field-A       Pic x(80)  Value Spaces.
01 Field-A-Len   Pic s9(9)  Comp  Value 0.
.....
Compute Field-A-Len = Function LENGTH(Field-A).
Perform varying Byte from 1 by 1 until Byte > Field-A-Len
   If Field-A(Byte:1) <> " "
     Move Field-A(Byte:) to Field-A(1:Field-A-Len)
     Move Field-A-Len to Byte
   End-If
End-Perform.
```


In-line PERFORMs - Timings

PERFORM paragraph timings*:

- CPU: 15.385; Wall: 15.528 (unopt)
- CPU: 15.419; Wall: 15.565 (opt)

In-Line PERFORM timings*:

- CPU: 14.253; Wall: 14.385 (unopt)
- CPU: 13.028; Wall: 13.147 (opt)

*Based on loop that left justifies a string using reference modification (see example in previous slide) for 1–million times. (917LX)

GoTo statement

- Considered "Taboo" in COBOL-land
- Go ahead – shoot me – I use them extensively to improve software performance without compromising structure
- Can **hurt** if used incorrectly – probably why considered "taboo" and not a "structured" option
- Used in other languages without problems

GoTo - Timings

PERFORM “GoTo-free” timings*:

- CPU: 0.753; Wall: 0.766 (unopt)
- CPU: 0.722; Wall: 0.735 (opt)

PERFORM “with GoTos” timings*:

- CPU: 0.377; Wall: 0.380 (unopt)
- CPU: 0.220; Wall: 0.222 (opt)

*Based on loop that performs a paragraph without “GoTo” statement versus another that uses “GoTo” statements to exit paragraph when conditions exist for 1–million times. (917LX)

Functions

- Available on all HP3000 systems for those who might move to another job and would like to use a specialized routine (how many date routines avail?)
- Supported by OS provider – reduced in-house maintenance
- Brings COBOL to the same realm as other languages
- SQRT, RANDOM, Date routines, Financial formulas, Table processing, etc.

File recommendations

- Flat files
- Ksam files – both CM & NM
- Temp files

Flat files

- Standard – serial read to find unique data
- RIO – random access
- CIRcular – keep last ‘nnn’ records – never gets full – like an odometer
- MSG – program to program communication (FIFO)
- Byte Stream – like DOS/PC files – each record is 1 byte

KSAM file - CM

- Keyed Sequential Access Method
- Requires 2 files – one data, one the key pointer file
- Use KSAMUTIL to build and maintain
- File System code – Compatibility Mode
- Temp or Perm
- Use if a key chain length is greater than 20,000 – 25,000 records

KSAM file - NM

- Keyed Sequential Access Method
- Requires 1 file – both data and key pointers together
- Use CI Commands to build and maintain
- File System code – Native Mode
- Temp or Perm
- DO NOT Use if a key chain length is greater than 20,000 – 25,000 records
  - 1.5+ million records w/ same key value – 23+ hours to load
  - 2 hours to load in CM KSAM

Temp or “New” files

- Bypasses Transaction Manager
- Load large files in temp domain then save as permanent, if needed


```
(917LX)                          (967)                          (979)
start timing "PERM" file         start timing "PERM" file       start timing "PERM" file
End timing "PERM" file           End timing"PERM" file          End timing "PERM" file
...CPU:141.466  Wall: 147.617    ...CPU: 62.792  Wall: 64.923   ...CPU: 13.697  Wall: 17.417

start timing "PERM w/Int" file   start timing "PERM w/Int" file start timing "PERM w/Int" file
End timing "PERM w/Int" file     End timing "PERM w/Int" file   End timing "PERM w/Int" file
...CPU:128.552  Wall: 135.534    ...CPU: 55.451  Wall: 57.530   ...CPU: 11.742  Wall: 16.840

start timing "TEMP" file         start timing "TEMP" file       start timing "TEMP" file
End timing of "TEMP" file        End of timing of "TEMP" file   End timing "TEMP" file
...CPU: 144.476  Wall: 150.877   ...CPU: 62.678  Wall: 64.812   ...CPU: 13.780  Wall: 16.562

start timing "TEMP w/Int" file   start timing "TEMP w/Int" file start timing "TEMP w/Int" file
End timing "TEMP w/Int" file     End timing "TEMP w/Int" file   End timing "TEMP w/Int" file
...CPU: 126.963  Wall: 132.980   ...CPU: 55.341  Wall: 57.426   ...CPU: 11.818  Wall: 14.214

start timing "NEW" file          start timing "NEW" file        start timing "NEW" file
End timing "NEW" file            End timing "NEW" file          End timing "NEW" file
...CPU: 145.858  Wall: 152.206   ...CPU: 63.221  Wall: 65.890   ...CPU: 13.725  Wall: 14.501

start timing "NEW w/Int" file    start timing "NEW w/Int" file  start timing "NEW w/Int" file
End timing "NEW w/Int" file      End timing "NEW w/Int" file    End timing "NEW w/Int" file
...CPU: 129.657  Wall: 135.685   ...CPU: 56.754  Wall:58.852    ...CPU: 11.669 Wall: 13.196
```


More before and after timings

- Unique key values in a flat file
  - Serially read flat file for every record processed
  - 4 hours to complete (“…worked fast in test”)
  - Copied code to several other new reports
  - Changed to temp Ksam file
  - 15 minutes to complete
- More yet to come

Speedware recommendations

- Use of DCLFILE directive
  - IMAGE data set – open mode
  - Change access to read-only
    (default – write access & can be slow)
  - Assign several names for same data set/file for different access requirements
- Be careful of the Optimizer – its choice versus your "coded" choice
  - Omnidex/Superdex first (can be a "gotcha")
  - IMAGE Key field second (use '[ ]' to force)
  - Serial read
  -

August 2001 COTC - T.RenzSoft 36

Powerhouse recommendations

- Send us your Powerhouse (Quiz, Quick, QTP, etc.) performance techniques, gotcha's, horror stories, and improvements realized

August 2001 COTC - T.RenzSoft 37

Suprtool recommendations

- Use and abuse of
  - CHAIN
  - $LOOKUP
  - Set Limits TableSize n
- Utilize TPI when appropriate via CHAIN
- Great use for totaling, summarizing, averaging, lead-in extracts, etc.

August 2001 COTC - T.RenzSoft 38

Suprtool - CHAIN

- Use when only a few or very small percentage is to be extracted
- No performance gain (misconception) with this read option (same as writing your own program)
- 0.5 – 1.0+ % of total set entries is a good cutoff point – if more, use GET and “IF $LOOKUP…”

Suprtool – CHAIN performance example

- Weekly job with ~50,000 +/- entries to read via CHAIN & TABLE
- 8+ data sets read w/ 1-2 million records each
- 8 – 12 hours to run (also a 27 hour hybrid job runs on request w/ 200,000+ entries)
- Changed job to use GET and “IF $LOOKUP…” for all extracts
- Run-time – 1½ hours (both versions)

Suprtool - $LOOKUP

- Used as a table lookup of a qualifying set of values when extracting data and using ‘IF’
- Best performance – must be last condition in the ‘IF’ statement
- Uses a binary search to locate table entries for each record read in file/data set
- Can decrease speed of extract if used as one of the first conditions on an ‘IF’ statement
- Have made changes to 3rd Party jobs and client jobs and improved extract performance by 20 – 50%

Suprtool - TableSize

- Default value of 2
- Increase when you have ample memory or nightly batch schedule
- Can reduce by day & increase by night via settings control file
- Increase number (up to 5) to inform MPE/iX to bring in more blocks into memory improving speed
- Can be degrading if system is overloaded or minimal memory causing large numbers of page faults

Suprtool - TPI

- Same rules apply as defined for CHAIN access
- Utilizes index items defined on data set
- "TPI" must be enabled
- Can use wild cards in data request
- Can only use one indexed item on CHAIN command

```
CHAIN DATA-SET,INDEXED-ITEM="A@", "DE"
CHain DATA-SET,INDEX-ITEM2="A@,-AB"
```


Suprtool – TPI example

- 3rd Party application
- Customer runs 5 iterations of job
- 30 minutes (average) to run – single threaded (2½ hour completion time)
- 3rd party solution – create separate job for extract and change other jobs to read extract
- Our solution – use CHAIN & TPI, use temp files and run at same time – run time 3 – 4 minutes per job (15 – 20 minute completion time)
- 3 years later – same run time average

Suprtool – other capabilities

- Unit Price average process:
  - 2-million part records
  - Written in 4GL and modified to only pull 3 per part
  - 7 – 8 hours weekly to complete
  - Modified to use Suprtool totally and average all records for each part
  - Takes 15 minutes to run

Suprtool – other capabilities (continued)

- Check data sets for corresponding entries
  - 4GL process to check that an entry exists in 8 corresponding data sets
  - Serially reads main data set – check key in all others
  - 6+ hours to complete
  - Modified serial read and checks to use Suprtool and SuprLink
  - Feeds to original 4GL process only those that are missing in corresponding data sets
  - Takes 25 – 30 minutes to complete

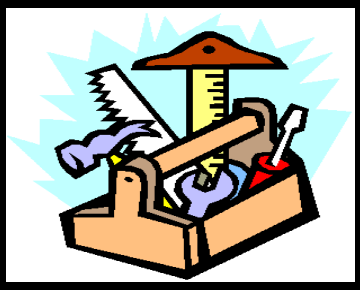
java

jvm performance

sh.hpbin.sys script	0.01
CI.PUB.SYS commandfile	0.10
Cognos Powerhouse	1.00
Perl	1.20
Basic/V Interpreted	1.64
Speedware	1.433 to 2.61
java -classic –nojit	**12.00**
java -hotspot –Xint	**21.00**
gcc unoptimized	106.00
Pascal (CM)	119.33
Pascal (NM, unopt)	130.10
Pascal (NM, Opt level 1)	155.50
c89 unoptimized	158.00
COBOL (NM, unoptimized)	167.00
COBOL (NM, opt level 1)	182.00
java -classic -jit	**194.00**
SPL (CM)	197.63
Basic/V Compiled	199.60
java -hotspot -Xmixed	**288.00**
Pascal (NM, Opt level 2)	379.90
gcc -O3	640.00
gcc -O3 -funroll-loops	667.00
c89 optimized	702.00

* Presented at HPWorld-2000 by Mike Yawn, HP – 09/13/2000
* Presentation title "java on mpe/ix: practical applications" (page/slide #6)
* Copied with permission from "jazz.external.hp.com/src/java/documentation/index.html"
* These numbers are estimates only; code written/contributed by various authors; not an actual measure of the various languages; numbers based on executing on a HPe3000 968; used same basic algorithm for number (SIEVE process); # times through loop in 10 seconds.

Toolbox



August 2001 COTC - T.RenzSoft 48

Performance Tools

- Glance/XL and Scope (old) – HP (OS performance)
- SOS and Performance Gallery – Lund Performance Systems (OS performance)
- SPT/XL – HP (view intrinsic calls by a process and their timings)
- DBGauge – TRenzSoft (application performance against your TurboIMAGE database and timings – open that "black box")
- Others?

August 2001 COTC - T.RenzSoft 49

Contact info

- Would love to hear, feature and share your performance findings and savings
- tom@trenzsoft.com
- www.trenzsoft.com
- 877-TRenzSoft (877-873-6976)

August 2001 COTC - T.RenzSoft 50

ΘΥΕΣΤΙΟΝΣ

August 2001 COTC - T.RenzSoft 51

Thank You!!

August 2001 COTC - T.RenzSoft 52

Other suggestions and recommendations

August 2001 COTC - T.RenzSoft 53